

#3
5-14-02
gm

Docket No.: 57454-180

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of

Toyohiko YOSHIDA

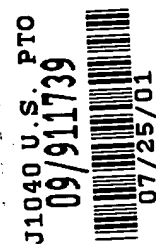
Serial No.:

Group Art Unit:

Filed: July 25, 2001

Examiner:

For: DATA PROCESSING DEVICE WITH INSTRUCTION TRANSLATOR AND
MEMORY INTERFACE DEVICE



**CLAIM OF PRIORITY AND
TRANSMITTAL OF CERTIFIED PRIORITY DOCUMENT**

Commissioner for Patents
Washington, DC 20231

Sir:


In accordance with the provisions of 35 U.S.C. 119, Applicant hereby claims the priority of:

Japanese Patent Application No. 2001-012687, filed January 22, 2001

cited in the Declaration of the present application. A Certified copy is submitted herewith.

Respectfully submitted,

MCDERMOTT, WILL & EMERY


Stephen A. Becker
Registration No. 26,527

600 13th Street, N.W.
Washington, DC 20005-3096
(202) 756-8000 SAB:prp
Date: July 25, 2001
Facsimile: (202) 756-8087

5145T-18U
YOSHIDA
July 25, 2001

日 本 国 特 許 庁
PATENT OFFICE
JAPANESE GOVERNMENT

McDermott, Will & Emery

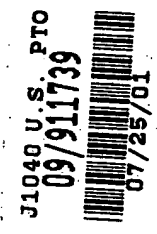
別紙添付の書類に記載されている事項は下記の出願書類に記載されて
いる事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed
with this Office.

出 願 年 月 日
Date of Application: 2001年 1月22日

出 願 番 号
Application Number: 特願2001-012687

出 願 人
Applicant(s): 三菱電機株式会社



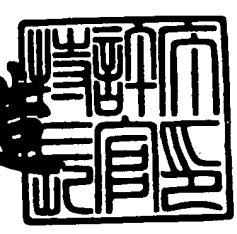
CERTIFIED COPY OF
PRIORITY DOCUMENT

CERTIFIED COPY OF
PRIORITY DOCUMENT

2001年 2月 9日

特許庁長官
Commissioner,
Patent Office

及川耕造



出証番号 出証特2001-3006755

【書類名】 特許願

【整理番号】 527420JP01

【提出日】 平成13年 1月22日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 9/30
G06F 9/455

【発明者】

【住所又は居所】 東京都千代田区丸の内二丁目2番3号 三菱電機株式会社
社内

【氏名】 吉田 豊彦

【特許出願人】

【識別番号】 000006013

【氏名又は名称】 三菱電機株式会社

【代理人】

【識別番号】 100064746

【弁理士】

【氏名又は名称】 深見 久郎

【選任した代理人】

【識別番号】 100085132

【弁理士】

【氏名又は名称】 森田 俊雄

【選任した代理人】

【識別番号】 100091409

【弁理士】

【氏名又は名称】 伊藤 英彦

【選任した代理人】

【識別番号】 100096781

【弁理士】

【氏名又は名称】 堀井 豊

【選任した代理人】

【識別番号】 100096792

【弁理士】

【氏名又は名称】 森下 八郎

【手数料の表示】

【予納台帳番号】 008693

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 命令トランスレータを備えたデータ処理装置およびメモリインタフェース装置

【特許請求の範囲】

【請求項 1】 プロセッサコアと、

前記プロセッサコアと、所定の外部メモリ空間にマップされる外部メモリとの間に設けられるメモリインタフェース部とを含み、

前記メモリインタフェース部は、

前記プロセッサコアから前記外部メモリ空間へのアクセスのためのアドレス値を受け、前記外部メモリから命令をフェッチするための命令フェッチ手段と、

前記外部メモリから受ける、前記プロセッサコアに対する非ネイティブ命令をネイティブ命令に変換するための変換器と、

前記プロセッサコアから前記外部メモリ空間へのアクセス時のアドレス値が所定領域内にあるか否かに依存して、前記外部メモリ空間から読出された命令と、前記外部メモリ空間から読み出された命令を前記変換器で変換した結果の命令とのいずれかを選択的に前記プロセッサコアに与えるための選択手段とを含む、命令トランスレータを備えたデータ処理装置。

【請求項 2】 前記命令フェッチ手段は、

前記プロセッサコアから前記外部メモリ空間へのアクセス時のアドレスに対して所定の変換を行なうためのアドレス変換手段と、

前記アドレスが、前記所定領域内にあるか否かにしたがって、前記プロセッサコアからのアドレスと、前記アドレス変換手段の出力するアドレスとのいずれかを選択的に前記外部メモリに与えるための手段とを含む、請求項 1 に記載の命令トランスレータを備えたデータ処理装置。

【請求項 3】 前記変換手段は、入力されたアドレス値を 2 の n 乗 (n は自然数) で除して出力する除算手段を含む、請求項 2 に記載の命令トランスレータを備えたデータ処理装置。

【請求項 4】 前記除算手段は、入力されたアドレス値を n ビット右シフトするシフタを含む、請求項 3 に記載の命令トランスレータを備えたデータ処理装

置。

【請求項 5】 プロセッサコアと、所定の外部メモリ空間にマップされる外部メモリとの間に設けられるメモリインタフェース装置であって、

前記プロセッサコアから前記外部メモリ空間へのアクセスのためのアドレス値を受け、前記外部メモリから命令をフェッチするための命令フェッチ手段と、

前記外部メモリから受ける、前記プロセッサコアに対する非ネイティブ命令をネイティブ命令に変換するための変換器と、

前記プロセッサコアから前記外部メモリ空間へのアクセス時のアドレス値が所定領域内にあるか否かに依存して、前記外部メモリ空間から読出された命令と、前記外部メモリ空間から読み出された命令を前記変換器で変換した結果の命令とのいずれかを選択的に前記プロセッサコアに与えるための選択手段とを含む、命令トランスレータを備えたメモリインタフェース装置。

【請求項 6】 前記命令フェッチ手段は、

前記プロセッサコアから前記外部メモリ空間へのアクセス時のアドレスに対して所定の変換を行なうためのアドレス変換手段と、

前記アドレスが、前記所定領域内にあるか否かにしたがって、前記プロセッサコアからのアドレスと、前記アドレス変換手段の出力するアドレスとのいずれかを選択的に前記外部メモリに与えるための手段とを含む、請求項 5 に記載の命令トランスレータを備えたメモリインタフェース装置。

【請求項 7】 前記変換手段は、入力されたアドレス値を 2 の n 乗 (n は自然数) で除して出力する除算手段を含む、請求項 6 に記載の命令トランスレータを備えたメモリインタフェース装置。

【請求項 8】 前記除算手段は、入力されたアドレス値を n ビット右シフトするシフタを含む、請求項 7 に記載の命令トランスレータを備えたメモリインタフェース装置。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

この発明はあるプロセッサにとっての非ネイティブ命令を、そのプロセッサの

ネイティブ命令に変換するための命令トランスレータを備え、非ネイティブ命令からなる処理ルーチンとネイティブ命令からなる処理ルーチンの混在したプログラムを高速に実行することが可能なデータ処理装置とそのためのメモリインタフェース装置とに関する。

【0002】

【従来の技術】

プロセッサアーキテクチャとそのプロセッサで実行可能な命令体系とは密接な関係を持つ。一方、プロセッサアーキテクチャが進化して命令体系が新しくなると、旧命令体系で実現されたプログラムコードはそのままでは実行不可能となることが通常である。どのようにして旧命令体系で実現されたプログラム資産を有効に承継するかが問題となる。そのため、ある命令体系を持つ新プロセッサで、旧命令体系にしたがって設計された旧プロセッサ用に記述されたプログラムを実行するようにするための手法が多く開発されている。

【0003】

旧プロセッサ用に記述されたプログラムを新プロセッサで実行するために従来行なわれている代表的な方法に、新プロセッサのハードウェアに旧プロセッサの機能を持たせる方法がある。図21を参照して、そうした方法を実現する従来のデータ処理装置500は、旧プロセッサの命令と新プロセッサの命令との双方をデコードする機能を有する多機能命令デコーダ5およびそれら命令を実行する機能を有する演算部6を備えたプロセッサ1と、プロセッサ1に接続されるバス4と、バス4に接続されるデータメモリ2および命令メモリ3とを含む。

【0004】

命令メモリ3は、旧プロセッサの命令と新プロセッサの命令との双方を保持する。多機能命令デコーダ5は、命令メモリ3から読出され、バス4を介してプロセッサ1に転送された命令をデコードする。このとき、多機能命令デコーダ5は、この命令が新プロセッサ用のものであっても、旧プロセッサ用のものであってもデコードする機能を有する。デコードされた命令は演算部6によって実行される。データメモリ2には、新プロセッサ用の命令からも、旧プロセッサ用の命令からもアクセスできる。

【 0 0 0 5 】

このように新プロセッサのハードウェアに旧プロセッサのハードウェアの機能を持たせる例は、たとえば「IA-32 Application Execution Model in an IA-64 System Environment」 (IA-64 Application Developer & Architecture Guide, Chapter 6, May 1999) に詳しく記載されている。

【 0 0 0 6 】

旧プロセッサ用に記述されたプログラムを新プロセッサで実行するための従来の他の方法として、旧プロセッサ用のソフトウェアを新プロセッサ用のソフトウェアに変換した後に実行する方法、および旧プロセッサ用の命令の動作を新プロセッサ用のソフトウェアでエミュレートする方法がある。この方法は例えば、Tom Thompsonによる「An Alpha in PC Clothing (Digital Equipment's new x86 emulator technology makes an Alpha system a fast x86 clone)」 (BYTE, pp. 195-196, February 1996) に詳しく記載されている。

【 0 0 0 7 】

ところで、ある命令体系で記述されたプログラムを他の命令体系にしたがって設計されたプロセッサで実行することは、これ以外の場合にも有効である。たとえば、ある命令体系のサブセットを定義してその縮小された命令体系でプログラムを記述すると、プログラムサイズを小さくすることができる。また、J A V ATM (Java及びすべてのJava関連の商標及びロゴは、米国及びその他の国における米国Sun Microsystems, Incの商標又は登録商標である。) 言語は、仮想的なプロセッサの命令体系を定めてプログラムを記述し、その同一のプログラムを複数のプロセッサでそれぞれのプロセッサの命令体系を用いて実行する仕組みとなっている。このためJ A V A言語で記述されたプログラムは、異なる命令体系の複数種類のプロセッサで共有して実行することができる。

【 0 0 0 8 】

プログラムサイズを小さくする目的でサブセットの縮小命令体系を作り、プロセッサの多機能命令デコーダで非縮小命令体系の命令と縮小命令体系の命令との双方の命令をデコードする方法は既に多数提案されている。たとえばJames L. T urleyによる「Thumb Squeezes ARM Code Size (New Core Module Provides Opti

mized Second Instruction Set)」 (Micro Processor Report, Vol. 9, No. 4, pp. 1, 6-9, March 27, 1995) の記載を参照。

【0009】

【発明が解決しようとする課題】

しかし、上述したような従来の手法には、いずれも以下に述べるような問題点がある。

【0010】

複数個の命令体系によって記述されたプログラムを実行する機能をプロセッサのハードウェアに持たせる場合、ハードウェアが複雑になり、かつそのサイズが大きくなる。また、実行すべき命令体系を追加したり変更したりする場合には、ハードウェア全体を設計し直す必要があり、柔軟に対応することが困難である。

【0011】

ソフトウェアによりプログラムを変換する場合には次のような問題がある。プログラム自体を変換する場合には、変換後プログラムを保持するために大容量のメモリを新たに必要とする。その結果、メモリのコストが増大しデータ処理装置のコストも上昇する。また命令の動作を他の命令体系の命令でエミュレートする場合には、演算結果をエミュレートすることが必要であることはもちろん、プログラムカウンタの値、および必要がある場合にはフラグまでエミュレートしなければならない。そのため、一つの命令の動作を、別の体系の多数の命令で置換することが必要になる。その結果、動作速度が大幅に低下するという問題がある。

【0012】

この発明はこうした問題を解決するためになされたもので、プロセッサのハードウェア自体は変更せずに、複数個の異なる命令体系の命令からなるプログラムをネイティブ命令を用いて高速に実行可能なデータ処理装置であって、大容量のメモリを必要としないデータ処理装置、およびそのための命令トランスレータを備えたメモリインターフェース装置を提供することを目的とする。

【0013】

【課題を解決するための手段】

請求項1に記載の発明にかかるデータ処理装置は、プロセッサコアと、プロセ

ッサコアと所定の外部メモリ空間にマップされる外部メモリとの間に設けられるメモリインタフェース部とを含み、メモリインタフェース部は、プロセッサコアから外部メモリ空間へのアクセスのためのアドレス値を受け、外部メモリから命令をフェッチするための命令フェッチ手段と、外部メモリから受ける、プロセッサコアに対する非ネイティブ命令をネイティブ命令に変換するための変換器と、プロセッサコアから外部メモリ空間へのアクセス時のアドレス値が所定領域内にあるか否かに依存して、外部メモリ空間から読出された命令と、外部メモリ空間から読み出された命令を変換器で変換した結果の命令とのいずれかを選択的にプロセッサコアに与えるための選択手段とを含む。

【0014】

プロセッサ外部では非ネイティブ命令およびネイティブ命令がバス上を転送されるが、メモリインタフェース部において非ネイティブ命令はネイティブ命令に変換される。そのためプロセッサコア内部ではネイティブ命令のみが転送される。プロセッサコアは実行中の命令がネイティブ命令か、非ネイティブ命令かを判断することなく命令を単純に実行することにより、非ネイティブ命令処理ルーチンとネイティブ命令処理ルーチンとが混在するプログラムを実行することができる。プロセッサコア内部ではネイティブ命令のみなので、実行も高速である。

【0015】

請求項2に記載の発明にかかるデータ処理装置は、請求項1に記載の発明の構成に加えて、命令フェッチ手段は、プロセッサコアから外部メモリ空間へのアクセス時のアドレスに対して所定の変換を行なうためのアドレス変換手段と、アドレスが、所定領域内にあるか否かにしたがって、プロセッサコアからのアドレスと、アドレス変換手段の出力するアドレスとのいずれかを選択的に外部メモリに与えるための手段とを含む。

【0016】

非ネイティブ命令の命令長とネイティブ命令の命令長とが異なる場合でも、プロセッサコアは命令長の違いを意識する必要がない。

【0017】

請求項3に記載の発明にかかるデータ処理装置は、請求項2に記載の発明の構

成に加えて、変換手段は、入力されたアドレス値を2の n 乗（ n は自然数）で除して出力する除算手段を含む。

【0018】

非ネイティブ命令の命令長がネイティブ命令の命令長を2の n 乗で除した大きさのとき、単純な回路だけで非ネイティブ命令をフェッチするためのアドレス変換を行なうことができる。

【0019】

請求項4に記載の発明にかかるデータ処理装置は、請求項3に記載の発明の構成に加えて、除算手段は、入力されたアドレス値を n ビット右シフトするシフタを含む。

【0020】

非常に単純な回路で非ネイティブ命令をフェッチするためのアドレス変換が行なえる。

【0021】

請求項5に記載の発明にかかるメモリインタフェース装置は、プロセッサコアと、所定の外部メモリ空間にマップされる外部メモリとの間に設けられるメモリインタフェース装置であって、プロセッサコアから外部メモリ空間へのアクセスのためのアドレス値を受け、外部メモリから命令をフェッチするための命令フェッチ手段と、外部メモリから受ける、プロセッサコアに対する非ネイティブ命令をネイティブ命令に変換するための変換器と、プロセッサコアから外部メモリ空間へのアクセス時のアドレス値が所定領域内にあるか否かに依存して、外部メモリ空間から読出された命令と、外部メモリ空間から読み出された命令を変換器で変換した結果の命令とのいずれかを選択的にプロセッサコアに与えるための選択手段とを含む。

【0022】

プロセッサ外部では非ネイティブ命令およびネイティブ命令がバス上を転送されるが、メモリインタフェース部において非ネイティブ命令はネイティブ命令に変換される。そのためプロセッサコア内部ではネイティブ命令のみが転送される。プロセッサコアは実行中の命令がネイティブ命令か、非ネイティブ命令かを判

断することなく命令を単純に実行することにより、非ネイティブ命令処理ルーチンとネイティブ命令処理ルーチンとが混在するプログラムを実行することができる。プロセッサコア内部ではネイティブ命令のみなので、プロセッサコアにおける実行を高速にすることができる。

【0023】

請求項6に記載の発明にかかるメモリインタフェース装置は、請求項5に記載の発明の構成に加えて、命令フェッチ手段は、プロセッサコアから外部メモリ空間へのアクセス時のアドレスに対して所定の変換を行なうためのアドレス変換手段と、アドレスが、所定領域内にあるか否かにしたがって、プロセッサコアからのアドレスと、アドレス変換手段の出力するアドレスとのいずれかを選択的に外部メモリに与えるための手段とを含む。

【0024】

非ネイティブ命令の命令長とネイティブ命令の命令長とが異なる場合でも、プロセッサコアは命令長の違いを意識する必要がない。

【0025】

請求項7に記載の発明にかかるメモリインタフェース装置は、請求項6に記載の発明の構成に加えて、変換手段は、入力されたアドレス値を2の n 乗（ n は自然数）で除して出力する除算手段を含む。

【0026】

非ネイティブ命令の命令長がネイティブ命令の命令長を2の n 乗で除した大きさのとき、単純な回路で非ネイティブ命令をフェッチするためのアドレス変換を行なうことができる。

【0027】

請求項8に記載の発明にかかるメモリインタフェース装置は、請求項7に記載の発明の構成に加えて、除算手段は、入力されたアドレス値を n ビット右シフトするシフタを含む。

【0028】

非常に単純な回路で非ネイティブ命令をフェッチするためのアドレス変換が行なえる。

【 0 0 2 9 】

【 発 明 の 実 施 の 形 態 】

図 1 を参照して、この発明の実施の形態のデータ処理装置は、半導体集積回路上に形成されたプロセッサ 5 1 0 と、プロセッサ 5 1 0 に接続されたアドレスバス 3 1、データバス 3 2 および制御バス 3 3 と、いずれもアドレスバス 3 1、データバス 3 2 および制御バス 3 3 に接続された、データメモリ 5 2 1、ネイティブ命令用メモリ 5 2 2、J A V A 命令メモリ 5 2 3、および R A M (ランダムアクセスメモリ) 5 2 0 とを含む。

【 0 0 3 0 】

図 2 を参照して、プロセッサ 5 1 0 は、半導体集積回路の 1 チップからなり、プロセッサのコア 1 0 0 と、命令メモリ 5 0 1 と、データメモリ 5 0 2 と、アドレスバス 3 1、データバス 3 2 および制御バス 3 3 に接続されたバスインターフェイス部 5 0 3 と、これらを互いに接続してアドレスおよびデータを送信するためのデータアドレスバス 1 0 6 およびデータバス 1 0 7 と、コア 1 0 0、命令メモリ 5 0 1 およびバスインターフェイス部 5 0 3 を相互に接続する 6 4 ビット幅の命令アドレスバス 1 0 4 および 6 4 ビット幅の命令バス 1 0 5 とを含む。

【 0 0 3 1 】

コア 1 0 0 は、V L I W (Very Long Instruction Word) 型命令体系を有するプロセッサコアである。コア 1 0 0 は、命令バス 1 0 5 から入力された V L I W 命令をデコードするための命令デコーダ 1 1 0 と、命令デコーダ 1 1 0 によってデコードされた命令を実行するためのメモリ演算部 1 3 0 および整数演算部 1 4 0 と、メモリ演算部 1 3 0 および整数演算部 1 4 0 に複数のバスで接続された、ソフトウェアで読み書きできる汎用レジスタファイル 1 2 0 とを含む。

【 0 0 3 2 】

命令デコーダ 1 1 0 は二つのサブ命令デコーダ 1 1 1 および 1 1 2 を含む。

メモリ演算部 1 3 0 は、アドレス演算器 1 3 1、P C (プログラムカウンタ) 演算器 1 3 2、シフタ 1 3 3 および A L U (Arithmetic Logic Unit) 1 3 4 などの演算器を含む。メモリ演算部 1 3 0 は、サブ命令デコーダ 1 1 1 の出力にしたがい、メモリアクセス命令、P C 制御命令、整数演算命令などを実行するため

のものである。整数演算部 1 4 0 は、シフタ 1 4 1、ALU 1 4 2、乗算器 1 4 3 および アキュムレータ 1 4 4 を含む。整数演算部 1 4 0 は、サブ命令デコーダ 1 1 2 の出力にしたがい整数演算命令を実行するためのものである。メモリ演算部 1 3 0 および整数演算部 1 4 0 は、2 つのサブ命令を並列に実行する場合と、それぞれ独立に 1 つのサブ命令を実行する場合とがある。

【0 0 3 3】

図 3 を参照して、プロセッサ 5 1 0 が有するレジスタファイル 1 2 0 は、6 4 本の 3 2 ビット汎用レジスタであるレジスタ 1 5 0 ~ 1 5 2、…、1 6 2、1 6 3 を含む。なお後述するようにレジスタ 1 6 3 は二通りに使い分けられるので、それらをそれぞれレジスタ 1 6 3 a および 1 6 3 b として便宜上別々のものとして示してある。プロセッサ 5 1 0 はさらに、制御レジスタ 1 7 0 ~ 1 8 0 を含む。また、図 2 に示すアキュムレータ 1 4 4 はそれぞれ 6 4 ビットの二つのアキュムレータ 1 4 4 a およびアキュムレータ 1 4 4 b を含む。

【0 0 3 4】

レジスタ 1 5 0 は常にゼロを保持するレジスタである。レジスタ 1 6 2 は非割込処理中のスタックトップのデータを保持するためのものである。レジスタ 1 6 3 b は非割込処理中のスタックポインタでスタックトップのすぐ下のデータのアドレスを保持するためのものである。

【0 0 3 5】

レジスタ 1 6 3 は、PSW（プロセッサステータスワード）である制御レジスタ 1 7 0 中にあるモードビットで切り替わり、割込処理中はレジスタ 1 6 3 a として使用され、非割込処理中はレジスタ 1 6 3 b として使用される。

【0 0 3 6】

制御レジスタ 1 7 0 ~ 1 8 0 は、それぞれ所定の要素のための専用のレジスタである。たとえば制御レジスタ 1 7 0 は PSW であって、演算により変化するフラグ、割込処理処理中か否か、割込マスク中か否か、デバック中か否かなど、プロセッサ 5 1 0 の動作モードを示すモードビットを含む。制御レジスタ 1 7 2 はプログラムカウンタ（PC）であり、現在実行中の命令のアドレスを示す。制御レジスタ 1 7 1 および 1 7 3 は、割込受付時、例外発生時、トラップ発生時にそ

れぞれ制御レジスタ 1 7 0 および 1 7 2 の値をコピーして保持するためのものである。そのほかの制御レジスタの役割に付いては図 3 に記載したとおりである。それらは本発明と大きな関連を持たないので、記載を簡単にする目的でここではそれらについての詳細な説明は行なわない。

【 0 0 3 7 】

アキュムレータ 1 4 4 a および 1 4 4 b は、乗算結果、積和演算結果を保持するためのものである。アキュムレータ 1 4 4 a および 1 4 4 b はそれぞれ、汎用レジスタの 2 倍のビット長である 6 4 ビットのデータを保持することができる。

【 0 0 3 8 】

図 4 を参照して、制御レジスタ 1 7 0 が保持する P S W は 3 2 ビットであって、割込処理中か非割込処理中かを示すモードビットである S M ビット 1 7 0 a と、割込許可中か割込禁止中かを示す I E ビット 1 7 0 b と、命令の実行条件を制御する F 0 ビット 1 7 0 c および F 1 ビット 1 7 0 d とを含む。この他に制御レジスタ 1 7 0 は、R P ビット、M D ビット、および F 2 ～ F 7 の各ビットを含む。これらの意味については図 4 に示す通りである。割込み許可中はプロセッサ 1 5 0 へ外部から割込みが要求されているとき、V L I W 命令の切れ目で割込みが受け付けられる。

【 0 0 3 9 】

図 5 を参照して、プロセッサ 5 1 0 のバスインターフェース部 5 0 3 は、J A V A バイトコードをネイティブ命令にハードウェア（以下単に「H W」と呼ぶ。）トランスレート、すなわちハードウェア回路により変換するためのトランスレーション回路 5 4 0 と、制御バス 3 3 に接続され、H W トランスレーションの制御およびプロセッサ 5 1 0 のメモリ 5 2 0 ～ 5 2 3 へのアクセスを制御するための、アドレスレジスタ 5 3 3 を有する制御部 5 3 2 と、命令アドレスバス 1 0 4 およびデータアドレスバス 1 0 6 にそれぞれ接続された二つの入力と、信号線 5 4 2 に接続された出力とを有するセクタ 5 3 1 と、信号線 5 4 2 上のアドレスを 8 で除して、すなわち 3 ビット右シフトして、信号線 5 4 3 上に出力するためのアドレス信号シフト回路 5 3 4 と、信号線 5 4 3 および 5 4 2 にそれぞれ接続された二つの入力と、信号線 5 4 5 に接続された出力とを有し、トランスレーシ

ョン回路 5 4 0 および制御部 5 3 2 から信号線 5 4 6 を介して与えられる信号により制御されるセクタ 5 3 5 とを含む。

【 0 0 4 0 】

バスインターフェース部 5 0 3 はさらに、トランスレーション回路の出力に接続された信号線 5 4 4 と、信号線 5 4 5 とにそれぞれ接続された二つの入力と、アドレスバス 3 1 の下位 2 4 ビットに接続された出力とを有するセクタ 5 3 8 と、信号線 5 6 2 と 4 バイト幅の信号線 5 4 1 とにそれぞれ接続された選択側の 2 端子と、4 バイト幅のデータバス 3 2 に接続された 1 端子とを有し、データバス 3 2 からの入力を信号線 5 4 1 または 5 6 2 のいずれかに選択的に出力し、また信号線 5 6 2 上のデータを選択してデータバス 3 2 上に出力する機能を有するセクタ 5 3 7 と、信号線 5 4 1 に接続された入力と信号線 5 6 0 に接続された出力とを有するバス幅変換回路 5 6 0 と、トランスレーション回路 5 4 0 からの出力線 5 3 9 およびバス幅変換回路 5 6 0 からの 8 バイト幅の信号線 5 6 3 にそれぞれ接続された二つの入力と、命令バス 1 0 5 に接続された出力とを有するセクタ 5 3 6 と、4 バイト幅の信号線 5 6 2 と 6 4 ビット幅のデータバス 1 0 7 とにそれぞれ接続された二つの入出力端子を有するバス幅変換回路 5 6 1 とを含む。

【 0 0 4 1 】

信号線 5 4 1 はトランスレーション回路 5 4 0 の入力にも接続されている。信号線 5 4 2 は、制御部 5 3 2 にも接続されている。

【 0 0 4 2 】

制御部 5 3 2 は、信号線 5 3 1 から受けるアドレス値が所定のアドレス領域内にあるかどうかを判断し、この判断に応じてセクタ 5 3 5、5 3 8 および 5 3 6 を制御する。ここで所定のアドレス領域とは後述する図 1 2 のアドレス領域 5 3 7 に相当する。制御部 5 3 2 は受取るアドレス値がその所定のアドレス領域内にあるとき、信号線 5 4 3 を選択して信号線 5 4 3 のアドレス値（シフト回路 5 3 4 により信号線 5 4 2 のアドレス値が変換されたもの）を信号線 5 4 5 に与え、そうでないときには、信号線 5 4 2 を選択して信号線 5 4 2 のアドレス値を信号線 5 4 5 に与えるようにセクタ 5 3 5 を制御する。また制御部 5 3 2 は、受

取るアドレス値がその所定のアドレス領域内にあるとき、信号線 5 3 9 を選択して信号線 5 3 9 の命令（トランスレーション回路 5 3 9 により非ネイティブ命令が変換された後のネイティブ命令）を信号線 1 0 4 に与え、そうでないときには信号線 5 6 3 を選択して信号線 5 4 1 の命令（ネイティブ命令）を信号線 1 0 4 に与えるようにセレクタ 5 3 6 を制御する。また制御部 5 3 2 は、受取るアドレス値がその所定のアドレス領域内にあるとき、信号線 5 4 4 を選択して信号線 5 4 4 のアドレス値を信号線 3 1 に与え、そうでないときには信号線 5 4 5 を選択して信号線 5 4 5 のアドレス値を信号線 3 1 に与えるようにセレクタ 5 3 8 を制御する。

【 0 0 4 3 】

またプロセッサコア 1 0 0 は、メモリをアクセスするとき、命令アドレスで指定される命令に対するものか、またはデータアドレスで指定されるデータに対するものかを示す制御信号を出力する。セレクタ 5 3 1 は、この制御信号によって制御され、命令に対するアクセスのときは信号線 1 0 4 を選択し信号線 1 0 4 のアドレス値を信号線 5 4 2 に与え、データに対するアクセスのときは信号線 1 0 6 を選択し信号線 1 0 6 のアドレス値を信号線 5 4 2 に与える。同様に、セレクタ 5 3 7 もこの制御信号によって制御され、命令に対するアクセスのときは信号線 5 4 1 を選択し信号線 3 2 を信号線 5 4 1 と電氣的に接続し、データに対するアクセスのときは信号線 5 6 2 を選択し信号線 5 6 2 を信号線 1 0 7 に接続する。

【 0 0 4 4 】

図 6 を参照して、トランスレーション回路 5 4 0 は、信号線 5 4 1 を介してメモリ 5 2 3 からデータバス 3 2 の幅にあわせて 4 バイト単位でフェッチした J A V A バイトコードを一時的に蓄えるための入力バッファメモリ 5 5 1 と、入力バッファメモリ 5 5 1 から与えられる J A V A バイトコード 1 バイトを変換して 8 バイトのネイティブ命令一つを出力する変換器 5 5 2 と、変換器 5 5 2 の出力、信号線 5 4 5 および 5 4 6 からのアドレス信号および制御信号を受け、変換器 5 5 2 から出力されたネイティブコードを一時蓄えて信号線 5 3 9 に出力するための出力バッファメモリ 5 5 3 と、信号線 5 4 4、5 4 5 および 5 4 6 に接続され

、セクタ535から信号線545を介して入力されたアドレスを管理するためのアドレス管理回路554とを含む。

【0045】

図7を参照して、プロセッサ510は、命令を以下のようにしてパイプライン処理する。プロセッサ510は、機能的には、メモリ演算部130と整数演算部140とで行なわれるサブ命令をそれぞれ実行するためのMUパイプ139およびIUパイプ149を含む。これらパイプはいずれも、命令フェッチステージ191、デコードおよびアドレス計算ステージ192、演算およびメモリアクセスステージ193およびライトバックステージ194からなる。

【0046】

命令フェッチステージ191は、命令をフェッチして命令デコーダ110中の命令レジスタ113に保持するステージである。デコードおよびアドレス計算ステージ192では、この命令がサブ命令デコーダ111、112でデコードされ、同時にレジスタファイル120（入力であることを示すためにレジスタファイル120aとする。）がアクセスされてオペランドおよびPCのアドレス計算が行なわれる。演算およびメモリアクセスステージ193では、整数演算およびデータメモリアクセス処理が行なわれる。ライトバックステージ194では、演算結果およびメモリからフェッチされたデータがレジスタファイル120（出力側であることを示すためにレジスタファイル120bとする。）に再び書込まれる。

【0047】

図8を参照して、プロセッサ510の命令200は2ウェイのVLIW型命令であり、図示されるようなフォーマットを有する。すなわち、命令200は、各サブ命令の実行順序と長いサブ命令を定義するFMフィールド201a、201bと、サブ命令を格納するLコンテナ205およびRコンテナ206と、各サブ命令の実行条件を指定する条件（CC）フィールド203および204とを含む。

【0048】

条件フィールド203および204は、制御レジスタ170であるPSW中の

フラグ F 0 および F 1 (F 0 ビット 1 7 0 c および F 1 ビット 1 7 0 d) の値に依存した条件を指定する。たとえば条件フィールド 2 0 3 が「 0 0 0 」のとき、L コンテナ 2 0 5 に含まれるサブ命令は無条件に実行される。条件フィールド 2 0 4 が「 1 0 1 」のとき、R コンテナ 2 0 6 に含まれるサブ命令は F 0 = 1 かつ F 1 = 1 のとき実行され、フラグ F 0 および F 1 (F 0 ビット 1 7 0 c および F 1 ビット 1 7 0 d) がそれ以外の値をとる場合は無効化される。

【 0 0 4 9 】

FM フィールド 2 0 1 a および 2 0 1 b は、L コンテナ 2 0 5 と R コンテナ 2 0 6 とに含まれるサブ命令を実行する場合の実行動作を指定する。実行動作としては 4 つある。1 番目は、L コンテナ 2 0 5 と R コンテナ 2 0 6 とに含まれるサブ命令を並列に実行する動作である。2 番目は、L コンテナ 2 0 5 のサブ命令をまず実行して次に R コンテナ 2 0 6 のサブ命令を実行する動作である。3 番目は 2 番目の動作の逆であり、R コンテナ 2 0 6 のサブ命令をまず実行して次に L コンテナ 2 0 5 のサブ命令を実行する動作である。4 番目は L コンテナ 2 0 5 と R コンテナ 2 0 6 とに分割して保持された 1 つの長いサブ命令を実行する動作である。すなわち、FM フィールド 2 0 1 a および FM フィールド 2 0 1 b の値によって、上述した 4 つの動作のいずれかが選択される。

【 0 0 5 0 】

図 9 を参照して、L コンテナ 2 0 5 および R コンテナ 2 0 6 に保持されるサブ命令は以下のようなフォーマットを有する。サブ命令は 2 8 ビット長の短い命令と 5 4 ビット長の長いサブ命令とに分類される。短いサブ命令は、フォーマット 2 1 1 ~ 2 1 7 に示される 7 種類のフォーマットを有する。短いサブ命令フォーマットの概略をいえば、ビット位置 0 - 9 で演算の種類が示され、ビット位置 1 0 - 2 7 で最大 3 つのオペランドが指定される。長いサブ命令はフォーマット 2 1 8 に示されるようにビット位置 0 - 9 で演算の種類が示され、ビット位置 1 0 - 5 3 で 3 2 ビット長の即値データを含む最大 3 つのオペランドが指定される。なお、長いサブ命令の 3 2 ビットの即値は V L I W 命令ビット位置 2 6 - 3 1、3 6 - 4 3、および 4 6 - 6 3 に保持される。

【 0 0 5 1 】

フォーマット211は、メモリアクセス演算（ロード／ストア演算）を行なうサブ命令のフォーマットである。フォーマット212は汎用レジスタに保持されたオペランド間の演算（OP演算）を行なうサブ命令のフォーマットである。フォーマット213-217は分岐演算を行なうサブ命令のフォーマットである。長いサブ命令のフォーマット218は、上記した3種類の演算全てに共通で使用される。

【0052】

サブ命令を図7に示すようにプロセッサ510でパイプライン処理する場合には、OP演算、ロード／ストア演算、分岐演算のサブ命令はそれぞれ図10に示す4段のパイプライン221-223によって示されるように4段のパイプラインステージで実行される。これら各ステージは、図6に示される4つのステージに対応する。

【0053】

FMフィールド201aおよび201bによりサブ命令の実行順序が指定されたときには、サブ命令はMUパイプ139およびIUパイプ149によって図11に示すようにパイプライン処理される。ここで、ストールステージ234-236は、FMフィールド201aおよび201bの値にしたがってサブ命令に順序をつけて実行する場合に、一方のサブ命令の実行を遅延させるために挿入されるパイプラインステージである。

【0054】

次に、プロセッサ510に対して定義されたサブ命令の一覧を示す。この一覧において、各サブ命令のニーモニックを大文字で、その処理内容を各ニーモニックの右側に、それぞれ示す。

【0055】

ロード／ストア命令

LDB	符号拡張してレジスタに1バイトをロード
LDBU	ゼロ拡張してレジスタに1バイトをロード
LDH	符号拡張してレジスタに半ワードをロード
LDHH	レジスタ上位に半ワードをロード

LDHU	ゼロ拡張してレジスタに半ワードをロード
LDW	レジスタに1ワードをロード
LD2W	2つのレジスタに2ワードをロード
LD4BH	符号拡張して4つの半ワードレジスタに4つのバイトをロード
LD4BHU	ゼロ拡張して4つの半ワードレジスタに4つのバイトをロード
LD2H	2つのレジスタに2つの半ワードをロード
STB	レジスタから1バイトをストア
STH	レジスタから半ワードをストア
STHH	レジスタ上位から半ワードをストア
STW	レジスタから1ワードをストア
ST2W	レジスタから2ワードをストア
ST4HB	4つの半ワードレジスタから4つのバイトをストア
ST2H	2つのレジスタから2半ワードをストア
MODDEC	5ビット即値によってレジスタ値をデクリメント
MODINC	5ビット即値によってレジスタ値をインクリメント

転送命令

MVFSYS	制御レジスタから汎用レジスタに転送
MVTSYS	汎用レジスタから制御レジスタに転送
MVFACC	アキュムレータから1ワードを転送
MVTACC	2つの汎用レジスタから2ワードをアキュムレータに転送

比較命令

CMPcc	比較 cc= EQ(000), NE(001), GT(010), GE(011), LT(100), LE(101), PS-both positive(110), NG-both negative(111)
CMPUcc	符号なし比較 cc=GT(010), GE(011), LT(100), LE(101)

算術演算命令

ABS	絶対値
ADD	加算
ADDC	キャリー付加算
ADDHppp	半ワード加算 ppp=LLL(000),LLH(001),LHL(010),LHH(011), HLL(100),HLH(101),HHL(110),HHH(111)
ADDS	第3オペランドの符号をレジスタRbに加算
ADDS2H	2つの半ワードに符号を加算
ADD2H	2対の半ワードを加算
AVG	正の無限大方向の丸めによる平均
AVG2H	正の無限大方向の丸めにより2対の半ワードを平均
JOINpp	2つの半ワードをジョイン pp=LL(00),LH(01),HL(10),HH(11)
SUB	減算
SUBB	ボロー付の減算
SUBHppp	半ワードの減算 ppp=LLL(000),LLH(001),LHL(010),LHH(011), HLL(100),HLH(101),HHL(110),HHH(111)
SUB2H	2対の半ワードの減算

論理演算命令

AND	論理AND
OR	論理OR
NOT	論理NOT
XOR	排他的論理OR
ANDFG	フラグの論理AND
ORFG	フラグの論理OR
NOTFG	フラグの論理NOT

XORFG フラグの排他的論理OR

シフト演算命令

SRA	右に算術シフト
SRAHp	右に半ワードを算術シフト p=L(0),H(1)
SRA2H	2つの半ワードを右に算術シフト
SRC	レジスタを接続して右にシフト
SRL	右に論理シフト
SRLHp	半ワードを右に論理シフト p=L(0),H(1)
SRL2H	2つの半ワードを右に論理シフト
ROT	右ローテイト
ROT2H	2つの半ワードを右ローテイト

ビット演算命令

BCLR	ビットをクリア
BNOT	ビットを反転
BSET	ビットをセット
BTST	ビットをテスト

分岐命令

BRA	分岐
BRATZR	ゼロなら分岐
BRATNZ	ゼロでなければ分岐
BSR	サブルーチンに分岐
BSRTZR	ゼロならサブルーチンに分岐
BSRTNZ	ゼロでなければサブルーチンに分岐
DBRA	遅延分岐
DBRAI	即値付遅延分岐
DBSR	サブルーチンへの遅延分岐

DBSRI	サブルーチンへの即値付遅延分岐
DJMP	遅延ジャンプ
DJMPI	即値付遅延分岐
DJSR	サブルーチンへの遅延ジャンプ
DJSRI	サブルーチンへの即値付遅延ジャンプ
JMP	ジャンプ
JMPTZR	ゼロならジャンプ
JMPTNZ	ゼロでなければジャンプ
JSR	サブルーチンに分岐
JSRTZR	ゼロならサブルーチンにジャンプ
JSRTNZ	ゼロでなければサブルーチンにジャンプ
NOP	ノーオペレーション

OS関連命令

TRAP	トラップ
REIT	例外、割込およびトラップからの復帰

DSP算術演算命令

MUL	乗算
MULX	拡張精度の乗算
MULXS	拡張精度での乗算および左へのシフト
MULX2H	拡張精度で 2 対の半ワードを乗算
MULHXpp	拡張精度で 2 つの半ワードを乗算 pp=LL(00),LH(01),HL(10),HH(11)
MUL2H	2 対の半ワードを乗算
MACd	乗算および加算 (d=0,1)
MACSd	乗算、左 1 ビットシフト、加算 (d=0,1)
MSUBd	乗算および減算 (d=0,1)
MSUBSd	乗算、左 1 ビットシフト、減算 (d=0,1)

SAT	サチュレート
SATHH	ワードオペランドを上位半ワードにサチュレート
SAHL	ワードオペランドを下位半ワードにサチュレート
SATZ	正数にサチュレート
SATZ2H	2つの半ワードを正数にサチュレート
SAT2H	2つの半ワードオペランドをサチュレート

繰返し命令

REPEAT	1ブロックの命令を繰返す
REPEATI	1ブロックの命令を即値付繰返す

デバッガサポート命令

DBT	デバッグトラップ
RTD	デバッグ割込およびトラップからの復帰

図 1 2 に、図 1 に示すプロセッサ 5 1 0 がアドレスバス 3 1、データバス 3 2 および制御バス 3 3 を介して外部の各メモリ 5 2 0 ~ 5 2 3 をアクセスする場合のアドレスマップを示す。図 1 2 に示されるように、メモリ 5 2 0 ~ 5 2 2 はそれぞれ、RAM 用アドレス領域 5 7 5 (H' 2220 0000-H' 223F FFF8)、データメモリ用アドレス領域 5 7 1 (H' 1000 0000-H' 17FF FFF8)、ネイティブ命令メモリ用アドレス領域 5 7 2 (H' 1800 0000-H' 1FFF FFF8) にマップされる。メモリ 5 2 3 は、命令コードの変換を行わずにリードライトする場合にはアドレス領域 5 7 4 (H' 2100 0000-H' 211F FFF8) にマップされ、命令コードの変換を伴ってリードする場合はアドレス領域 5 7 3 (H' 2000 0000-H' 20FF FFF8) にマップされる。

【 0 0 5 6 】

プロセッサ 5 1 0 から見ると、メモリ 5 2 3 のアドレスは図 1 2 に示すアドレス領域 5 7 3 およびアドレス領域 5 7 4 となる。アドレス領域 5 7 4 でメモリ 5 2 3 をアクセスする場合、2 MB の空間の全バイト位置に有効なメモリがある。

一方、アドレス領域 5 7 3 でメモリ 5 2 3 をアクセスする場合、1 6 MB の空間の整地された各 8 バイトの上位 1 バイトには有効なメモリがあるが、下位 7 バイトにはメモリは存在しない。

【0 0 5 7】

なお、図 5 を参照して述べたことから明らかなように、外部のメモリは 4 バイトに整地されたアドレスによりアクセスされる。そのため外部のデータバス 3 2 は 4 バイト幅（3 2 ビット幅）となる。したがって、アドレス領域 5 7 2 などアクセスして命令メモリ 5 2 2 から一つのネイティブ命令（8 バイトの V L I W 命令）を読む場合、そのネイティブ命令のアクセスは 2 回に分けて行なわれる。具体的には、プロセッサがアドレスバス 3 1 に 2 回連続してアドレスを出力することにより、データバス 3 2 から 4 バイトのデータが 2 回連続して受取られる。バス幅変換回路 5 6 0 は、4 バイト幅の信号線 5 4 1 から受ける 4 バイトのデータを 8 バイトのデータに変換して 8 バイト幅の信号線 5 6 3 に転送するためのものである。すなわち、バス幅変換回路 5 6 0 は、1 回目の 4 バイトデータを信号線 5 4 1 から受取り一旦保持し、2 回目の 4 バイトデータを信号線 5 4 1 から受取ると、1 回目の 4 バイトデータと合わせた 8 バイトデータをネイティブ命令として出力する。

【0 0 5 8】

バス幅変換回路 5 6 1 の機能も同様である。すなわち、アドレス領域 5 7 1 などアクセスして 8 バイトのデータを読出または書込みする場合、4 バイト幅の信号線 5 6 2 から 2 回の連続したメモリアクセスにより受ける 2 つの 4 バイトデータを 1 つの 8 バイトデータにしてデータバス 1 0 7 に出力し、また、データバス 1 0 7 から受ける 1 つの 8 ビットデータを 2 つの 4 ビットデータに分けて 2 回の連続したメモリアクセスにより信号線 5 6 2、セクタ 5 3 7 を介してデータバス 3 2 に出力する。

【0 0 5 9】

なお、もしデータバス 3 2 のバス幅が 6 4 ビットであれば、バス幅変換回路 5 6 0、5 6 1 は不要である。この場合、信号線 5 4 1 および 5 6 3 は直接に接続される。同様に信号線 5 6 2 およびデータバス 1 0 7 は直接に接続される。

【0060】

上に構造を述べてきたプロセッサ510は以下のように動作する。最初に命令またはデータのアクセスがアドレス空間571～572、574～575に対して行なわれるHWトランスレーションなしのメモリアクセス時について説明する。この場合、図1を参照して、制御バス33上の信号により、アクセス対象となるメモリのチップセレクト信号をアサートする。図5に示すセクタ535および538で信号線542上の信号を選択してアドレスバス31にアドレスの下位24ビットを出力する。このとき、アクセスが命令リードか、データリードか、またはデータライトかに依存して、セクタ536～538が制御され命令バス105、データバス107およびデータバス32の接続関係とデータの流れが制御される。その結果、トランスレーション回路540の出力ではなく、セクタ537の出力がセクタ536によって選択され命令として信号線105を介してプロセッサコア100に与えられる。

【0061】

命令アクセスがアドレス空間573に行なわれる、HWトランスレーション付きのメモリアクセス時は、制御部532が信号線546を介してトランスレーション回路540、セクタ535および536を制御し、メモリ523からフェッチしたJ A V Aバイトコードをネイティブ命令に変換して命令バス105上に出力する。このとき、命令アドレスバス104からセクタ531を介して信号線542に、さらにアドレス信号シフト回路534に入力されたアドレス値は、アドレス信号シフト回路534によって3ビット右にシフトされる。すなわちアドレス値は1/8の値となって信号線545上に出力され、トランスレーション回路540に入力される。

【0062】

図6を参照して、トランスレーション回路540のアドレス管理回路554は、信号線545を介して入力された、アドレス領域573のアドレス値をアドレス領域574にマップしなおして信号線544上に出力する。この信号が図5に示されるセクタ538を介してアドレスバス31上に出力される。その結果メモリ523がアクセスされ、J A V Aバイトコードがフェッチされる。

【 0 0 6 3 】

このときのトランスレーション回路 5 4 0 の動作の詳細を図 6 を参照して説明する。アドレス管理回路 5 5 4 は、信号線 5 4 5 から入力されたアドレス値を、直前に入力されたアドレス値と比較し、対応する J A V A バイトコードが入力バッファメモリ 5 5 1 内に存在しているか判定する。もし対応する J A V A バイトコードが存在しているときにはその J A V A バイトコードを入力バッファメモリ 5 5 1 から変換器 5 5 2 に出力させる。

【 0 0 6 4 】

対応する J A V A バイトコードが入力バッファメモリ 5 5 1 内に存在していない場合、アドレス管理回路 5 5 4 は、信号線 5 4 5 を介して入力された、アドレス領域 5 7 3 の下位 2 4 ビットのアドレス値をアドレス領域 5 7 4 の下位 2 4 ビットのアドレス値としてメモリ 5 2 3 をアクセスするための制御信号を制御信号線 5 4 6 上に出力する。この際、J A V A バイトコードを整地された 4 バイト単位にフェッチするため、アドレス管理回路 5 5 4 から信号線 5 4 4 上に出力される 2 4 ビットのアドレス値の下位 2 ビットは、入力されたアドレス値にかかわらず常に 0 である。このアドレス値は図 5 に示すようにセレクタ 5 3 8 によって選択され、アドレスバス 3 1 上に出力される。これにより、J A V A バイトコードは 4 バイト単位でメモリ 5 2 3 からデータバス 3 2 を通して入力バッファメモリ 5 5 1 にフェッチされる。

【 0 0 6 5 】

図 1 3 を参照して、命令コード変換部 3 7 0 が変換する J A V A 命令と V L I W 命令との関係は以下のとおりである。たとえば 1 バイトの J A V A 命令 4 0 1 は 8 バイトの V L I W 命令 4 1 1 に変換される。J A V A 命令 4 0 1 は 1 バイトなので、この J A V A 命令 4 0 1 は J A V A 命令用メモリ 5 2 3 を 1 回アクセスするだけで読出すことができる。変換後の V L I W 命令 4 1 1 は出力バッファ 5 5 3 を経由して信号線 5 3 9 上に出力される。今度はセレクタ 5 3 6 は、アドレスがアドレス領域 5 7 3 の内部であるので、信号線 5 3 9 上の命令を選択してプロセッサコア 1 0 0 に与える。

【 0 0 6 6 】

2 バイトの J A V A 命令 4 0 2 は 2 つの 8 バイトの V L I W 命令 4 1 2 a、4 1 2 b に変換される。5 バイトの J A V A 命令 4 0 4 は 3 つの 8 バイトの V L I W 命令 4 1 4 a、4 1 4 b および 4 1 4 c に変換される。J A V A 命令 4 0 3、J A V A 命令 4 0 6 ~ 4 0 8 などについても同様で、J A V A 命令 4 0 3 は V L I W 命令 4 1 3 に、J A V A 命令 4 0 6 は V L I W 命令 4 1 6 に、J A V A 命令 4 0 7 は V L I W 命令 4 1 7 に、J A V A 命令 4 0 8 は 3 つの 8 バイトの V L I W 命令 4 1 8 a ~ 4 1 8 c に、それぞれ変換される。

【 0 0 6 7 】

図 1 4 に、コア 1 0 0 がメモリ領域 5 7 3 のアドレスで命令アクセス要求をしてからメモリ 5 2 3 がアクセスされて J A V A バイトコードがフェッチされ、ネイティブ命令に変換されてコア 1 0 0 へ転送される間のデータの流れを示す。コア 1 0 0 がメモリ領域 5 7 3 のアクセスを行なうときは、これは必ずネイティブ命令のフェッチである。したがってコア 1 0 0 は常に 8 バイトに整地されたアドレスをバスインターフェース部 5 0 3 に出力する。

【 0 0 6 8 】

メモリ領域 5 7 3 のアドレスはバスインターフェース部 5 0 3 でメモリ領域 5 7 4 のアドレスとして出力されメモリ 5 2 3 に対するアクセスが行なわれる。メモリ領域 5 7 3 は仮想的な J A V A バイトコード領域である。そのため、実際のメモリ 5 2 3 をアクセスしてフェッチされた J A V A バイトコード（非ネイティブ命令）はバスインターフェース部 5 0 3 でネイティブ命令に変換されてバス 1 0 5 からコア 1 0 0 へ転送される。つまり、コア 1 0 0 から見るとメモリ領域 5 7 3 の 1 6 M B のネイティブ命令が 1 / 8 に圧縮され、メモリ領域 5 7 4 の 2 M B の J A V A バイトコードとしてメモリ 5 2 3 に保存されている。そしてバスインターフェース部 5 0 3 がメモリ 5 2 3 から 4 バイト単位でフェッチした J A V A バイトコードを一旦入力バッファ 5 5 1 に蓄え、さらに入力バッファ 5 5 1 から変換器 5 5 2 に読出してネイティブ命令への変換を行なって出力バッファメモリ 5 5 3 を介してコア 1 0 0 に出力することで命令コードを伸長する機能を実現する。

【 0 0 6 9 】

なお、アドレス領域 5 7 4 をアクセスするときは、プロセッサ 5 1 0 は J A V A 命令メモリ 5 2 3 に記憶された J A V A バイトコードをデータとしてアクセスすることができる。したがってフェッチすべき J A V A バイトコードを指定するアドレスはデータアドレスバス 1 0 6 を介してアドレスバス 3 1 に転送され、J A V A バイトコードはデータバス 3 2 を介してメモリ 5 2 3 からデータバス 1 0 7 へ転送される。プロセッサ 5 1 0 は、「ソフトウェアインタプリタ」と呼ばれるネイティブ命令で記述されたソフトウェアを用いて、データとしてフェッチした J A V A バイトコードを実行することができ、または「ソフトウェアトランスレータ」と呼ばれるネイティブ命令で記述されたソフトウェアを用いてデータとしてフェッチした J A V A バイトコードをネイティブ命令に変換し、その後実行することができる。したがって本実施の形態によるプロセッサ 5 0 1 は、ハードウェアトランスレータ、ソフトウェアトランスレータ、およびソフトウェアインタプリタのいずれの方法によっても、J A V A バイトコードのような非ネイティブ命令で記述されたプログラムを実行できるようサポートされている。

【 0 0 7 0 】

なおプロセッサがどのような基準にしたがってハードウェアトランスレータ、ソフトウェアトランスレータ、およびソフトウェアインタプリタのいずれかを選択して J A V A バイトコードを実行するかについては、たとえば 2 0 0 0 年 1 2 月 4 日出願の、本願出願人と同一出願人による特願 2 0 0 0 - 3 6 8 7 2 9 号において検討されている。

【 0 0 7 1 】

図 1 5 ～ 図 2 0 は、変換器 3 7 2 で J A V A 命令を V L I W 命令に変換したときの具体的な例を示す。なお、J A V A 命令については Tim Lindholm and Frank Yellin, "The Java Virtual Machine Specification Second Edition," Sun Microsystems, Inc., 1999 に詳しく記載されている。

【 0 0 7 2 】

図 1 5 を参照して、1 バイトの J A V A 命令「iadd」はサブ命令「LDW R61, @(R63+, R0)」とサブ命令「ADD R62, R62, R61」とを順に実行する 1 つの V L I W 命令に変換される。「iadd」はスタックトップから 1 番目のデータと 2 番目の

データとである2つの32ビット整数を加算してスタックに書き戻すJ A V A命令である。プロセッサ510ではスタックトップのデータはレジスタR62に配置され、レジスタR63がスタックトップの次のデータのアドレスを示す。したがって、プロセッサ510では、J A V A命令「iadd」のオペレーションを、スタックトップから2番目の32ビットデータをレジスタR61にロードしてレジスタR62を4インクリメントするオペレーションを行なうサブ命令「LDW R61, @(R63+, R0)」と、レジスタR62とレジスタR61との2つの32ビット整数を加算してその結果をレジスタR62に書込むオペレーションを行なうサブ命令「ADD R62, R62, R61」とでエミュレートすることができる。このとき、P C値については、1つのV L I W命令を実行したことによりプロセッサ510のP C値を8番地進めることで、J A V A命令「iadd」の実行に対応してP C値を1進めることをエミュレートする。

【0073】

図16を参照して、2バイトのJ A V A命令「iload」は、1つのサブ命令「ADD/CN R50, #(0||vindex)」を持つV L I W命令と、サブ命令「STW R62, @(R63-, R4)」とサブ命令「LDW R62, @(R10, R50)」とを順次に実行するV L I W命令に変換される。「iload」はローカル変数領域から32ビット整数をフェッチしてスタックトップに保存するJ A V A命令である。プロセッサ510ではこれを、レジスタR50にローカル変数のインデックス値をロードするサブ命令「ADD/CN R50, #(0||vindex)」と、レジスタR62をスタックトップから2番目にブッシュするサブ命令「STW R62, @(R63-, R4)」と、スタックトップであるレジスタR62にローカル変数領域からデータをロードするサブ命令「LDW R62, @(R10, R50)」とに分解することによりシミュレートする。

【0074】

なお、ここでレジスタR4には、値「-4」を保持し、レジスタR10にはローカル変数領域のベースアドレスを保持する。P C値については2つのV L I W命令を実行したことによりプロセッサ510のP C値を16番地進めることで、J A V A命令「iload」の実行に対応してP C値を2番地進めることをシミュレートする。

【 0 0 7 5 】

図 1 7 を参照して、3 バイトの J A V A 命令「ifeq」は、サブ命令「ADD R62, R61, R0」とサブ命令「NOP」とを並列に実行する V L I W 命令と、サブ命令「LDW R62, @(R63+, R0)」とサブ命令「NOP」とを並列に実行する V L I W 命令と、1 つのサブ命令「BRATZR/CN R62, #(s||branchbyte1||branchbyte2)」を実行する V L I W 命令とに分解される。命令「ifeq」はスタックトップのデータが「0」なら分岐する J A V A 命令である。プロセッサ 5 1 0 ではこれを、レジスタ R 6 2 をレジスタ R 6 1 にコピーするサブ命令「ADD R62, R61, R0」と、スタックトップから 2 番目のデータをレジスタ R 6 2 にポップするサブ命令「LDW R62, @(R63+, R0)」と、レジスタ R 6 1 がゼロなら分岐するサブ命令「BRATZR/CN R62, #(s||branchbyte1||branchbyte2)」とに分解し、これら 3 つのサブ命令と 2 つの NOP 命令とを組合せて、計 3 つの V L I W 命令でシミュレートする。P C 値については、3 つの V L I W 命令を実行したことによりプロセッサ 5 1 0 の P C 値を 2 4 番地進めることで、J A V A 命令「ifeq」の実行に対応して P C 値を 3 番地進めることをシミュレートする。

【 0 0 7 6 】

図 1 8 を参照して、5 バイトの J A V A 命令「jsr_w」は、1 つのサブ命令「OR R10, #(branchbyte1||branchbyte2||branchbyte3||branchbyte4)」からなる V L I W 命令と、サブ命令「STW R62, @(R63-, R4)」とサブ命令「BSR R10」とを順次実行する V L I W 命令と、サブ命令「BRA #3」とサブ命令「NOP」とを並列に実行する V L I W 命令とに変換される。「jsr_w」は戻り先アドレスをスタックにプッシュして 4 バイトで指定されたアドレスのサブルーチンへジャンプする J A V A 命令である。プロセッサ 5 1 0 ではこの命令を、ジャンプ先アドレスをレジスタ R 1 0 にロードするサブ命令「OR R10, #(branchbyte1||branchbyte2||branchbyte3||branchbyte4)」と、スタックトップのレジスタ R 6 2 の値をスタックトップから 2 番目にプッシュするサブ命令「STW R62, @(R63-, R4)」と、戻り先アドレスをスタックトップであるレジスタ R 6 2 に保存してレジスタ R 1 0 で指定されたアドレスのサブルーチンへジャンプするサブ命令「JSR R10」と、サブルーチンから復帰した後に 2 つの V L I W 命令をスキップするための分

岐を行なうサブ命令「BRA #3」と、サブ命令「NOP」とに分解してシミュレートする。

【0077】

PC値については、3つのVLIW命令を実行し、2つのVLIW命令をスキップする分岐を行なうことによりプロセッサ510のPC値を40番地進めることで、JAVA命令「jsr_w」の実行に対応してPC値を5番地進めることをシミュレートする。

【0078】

図19と図20とは、複雑なJAVA命令を、VLIW命令からなりかつJAVA命令の機能を実行するサブルーチンをコールするVLIW命令に変換する例を示す。図19を参照して、この例では、浮動小数点数の加算を行なうJAVA命令「fadd」を、スタックトップのレジスタR62の値をスタックトップから2番目にプッシュするサブ命令「STW R62, @(R63-, R4)」と、戻り先アドレスをスタックトップであるレジスタR62に保存して#faddで指定されたアドレスのサブルーチンにジャンプするサブ命令「JSR #fadd」とを順次に行なうVLIW命令に変換する。プロセッサ510では、「fadd」のオペレーションをサブルーチン中に行ない、PC値の更新に関しては、一つのVLIW命令を実行することでプロセッサ510のPC値を8番地進めて「fadd」に対応するPC値を1番地進めることをシミュレートする。

【0079】

図20に示す例では、テーブルジャンプを行なうJAVA命令「tableswitch」を、スタックトップのレジスタR62の値をスタックトップから2番目にプッシュするサブ命令「STW R62, @(R63-, R4)」と、戻り先アドレスをスタックトップであるレジスタR62に保存して#tableswitchで指定されたアドレスのサブルーチンへジャンプするサブ命令「JSR #tableswitch」とを順次に行なうVLIW命令に変換する。プロセッサ510では、「tableswitch」のオペレーションとPC値の更新との両方をサブルーチン中でJAVA命令「tableswitch」で指定された各種パラメータをアクセスしてシミュレートする。このとき、プロセッサ510は、JAVA命令用（直結）アドレス領域125aでJAVA命令用の

トランスレータ付メモリ 25 をアクセスして J A V A 命令「tableswitch」で指定された各種パラメータをデータとして読出す。

【0080】

今回開示された実施の形態はすべての点で例示であって制限的なものではないと考えられるべきである。本発明の範囲は上記した説明ではなくて特許請求の範囲によって示され、特許請求の範囲と均等の意味および範囲内でのすべての変更が含まれることが意図される。

【0081】

【発明の効果】

請求項 1 に記載の発明によれば、非ネイティブ命令処理ルーチンとネイティブ命令処理ルーチンとが混在するプログラムを高速に実行することができる。

【0082】

請求項 2 に記載の発明によれば、請求項 1 に記載の発明の効果に加えて、非ネイティブ命令の命令長とネイティブ命令の命令長とが異なる場合でも、プロセッサコアは命令長の違いを意識する必要がなく、非ネイティブ命令処理ルーチンとネイティブ命令処理ルーチンとが混在するプログラムを実行することが可能となる。

【0083】

請求項 3 に記載の発明によれば、請求項 2 に記載の発明の効果に加えて、非ネイティブ命令の命令長がネイティブ命令の命令長を 2 の n 乗で除した大きさのとき、単純な回路だけで非ネイティブ命令をフェッチするためのアドレス変換を行なうことができる。

【0084】

請求項 4 に記載の発明によれば、請求項 3 に記載の発明の効果に加えて、非常に単純な回路で非ネイティブ命令をフェッチするためのアドレス変換が行なえる。

【0085】

請求項 5 に記載の発明によれば、このメモリインタフェースを有するプロセッサコアは実行中の命令がネイティブ命令か、非ネイティブ命令かを判断すること

なく命令を単純に実行することにより、非ネイティブ命令処理ルーチンとネイティブ命令処理ルーチンとが混在するプログラムを実行することができる。プロセッサコア内部ではネイティブ命令のみなので、プロセッサコアにおける実行を高速にすることができる。

【0086】

請求項6に記載の発明によれば、請求項5に記載の発明の効果に加えて、非ネイティブ命令の命令長とネイティブ命令の命令長とが異なる場合でも、プロセッサコアは命令長の違いを意識する必要がなく、プロセッサの構成に変更を加えずに非ネイティブ命令の処理ルーチンとネイティブ命令の処理ルーチンとが混在したプログラムを実行することができる。

【0087】

請求項7に記載の発明によれば、請求項6に記載の発明の効果に加えて、非ネイティブ命令の命令長がネイティブ命令の命令長を2のn乗で除した大きさのとき、単純な回路で非ネイティブ命令をフェッチするためのアドレス変換を行なうことができる。

【0088】

請求項8に記載の発明によれば、請求項7に記載の発明の効果に加えて、非常に単純な回路で非ネイティブ命令をフェッチするためのアドレス変換が行なえる。

【図面の簡単な説明】

【図1】 本発明の1実施の形態にかかる命令トランスレータ機能付メモリを備えたデータ処理装置のブロック図である。

【図2】 図1に示すプロセッサ510のブロック図である。

【図3】 プロセッサ510が有するレジスタの一覧を表形式で示す図である。

【図4】 プロセッサ510の制御レジスタ170の詳細を示す図である。

【図5】 プロセッサ510のバスインターフェース部503のブロック図である。

【図6】 バスインターフェース部503のトランスレーション回路540

のブロック図である。

【図 7】 プロセッサ 5 1 0 のパイプライン処理機構を説明するための図である。

【図 8】 プロセッサ 5 1 0 で実行可能な V L I W 命令のフォーマットを示す図である。

【図 9】 プロセッサ 5 1 0 で実行可能な V L I W 命令のサブ命令のフォーマットを示す図である。

【図 1 0】 プロセッサ 5 1 0 のサブ命令のパイプライン処理方法を示すための図である

【図 1 1】 プロセッサ 5 1 0 で 2 つのサブ命令をパイプライン処理する方法を説明するための図である。

【図 1 2】 図 1 に示すデータ処理装置のメモリマップを示す図である。

【図 1 3】 J A V A 命令を 1 または複数の V L I W 命令に変換するときの命令間の対応を示す図である。

【図 1 4】 メモリ 5 2 3 から J A V A バイトコードがフェッチされ、ネイティブ命令に変換されてコア 1 0 0 へ転送される間のデータの流れを示す図である。

【図 1 5】 J A V A 命令を V L I W 命令に変換する具体例を示す図である

【図 1 6】 J A V A 命令を V L I W 命令に変換する具体例を示す図である

【図 1 7】 J A V A 命令を V L I W 命令に変換する具体例を示す図である

【図 1 8】 J A V A 命令を V L I W 命令に変換する具体例を示す図である

【図 1 9】 J A V A 命令を V L I W 命令に変換する具体例を示す図である

【図 2 0】 J A V A 命令を V L I W 命令に変換する具体例を示す図である

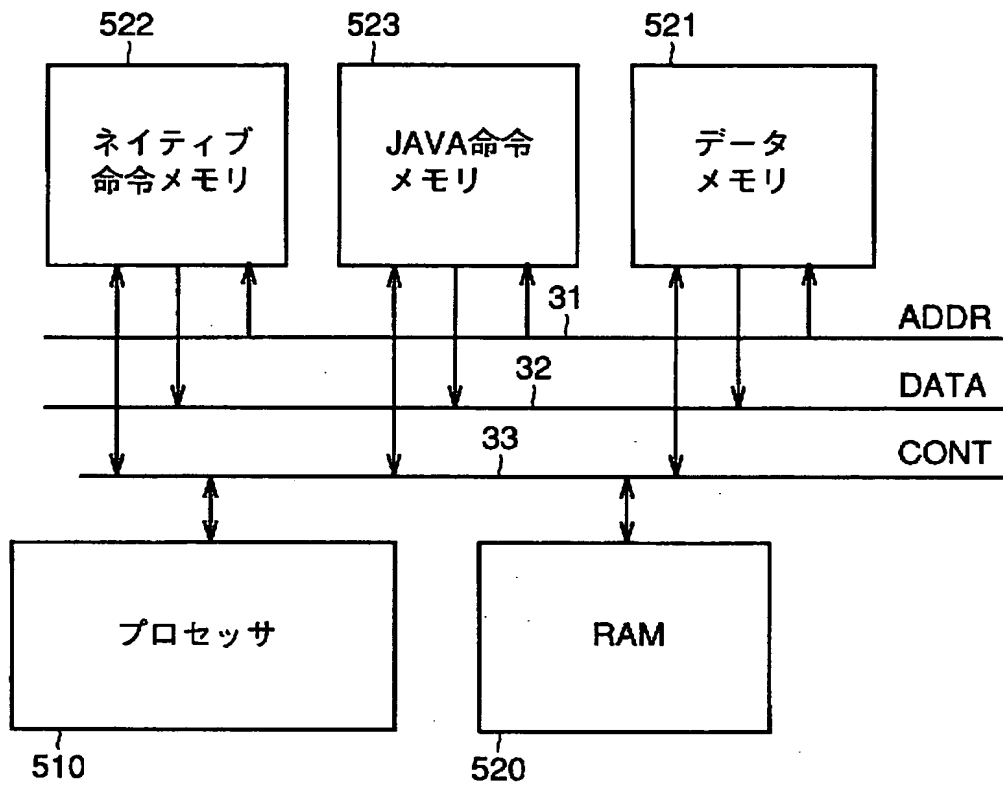
【図 2 1】 従来の旧命令エミュレーション機能付データ処理装置の概略ブロック図である。

【符号の説明】

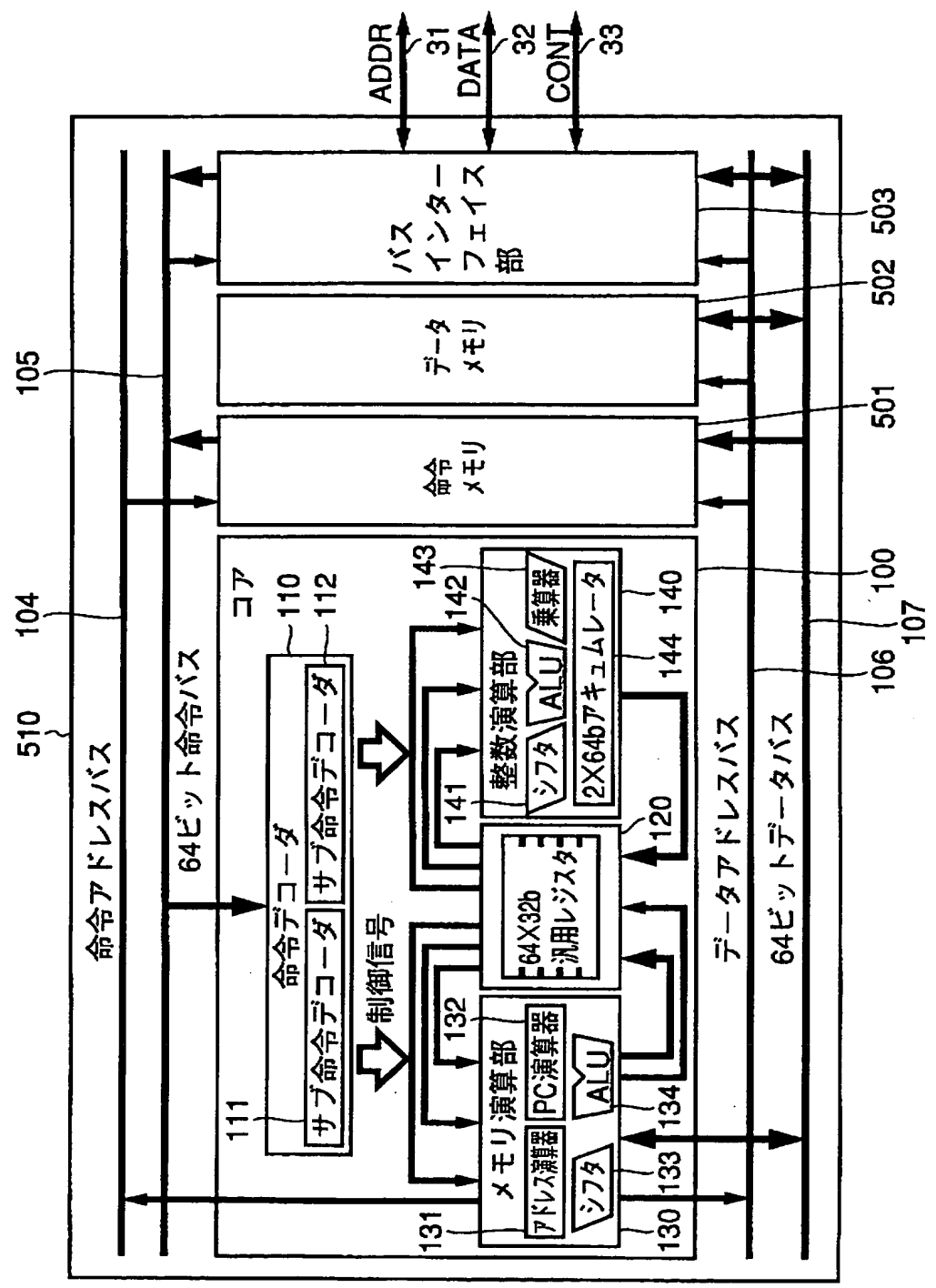
5 1 0 プロセッサ、5 2 0 RAM、5 2 1 データメモリ、5 2 2 ネイティブ命令メモリ、5 2 3 JAVA命令メモリ、5 0 3 バスインターフェース部、5 4 0 トランスレーション回路、5 5 4 アドレス管理回路、5 5 2 変換器。

【書類名】 図面

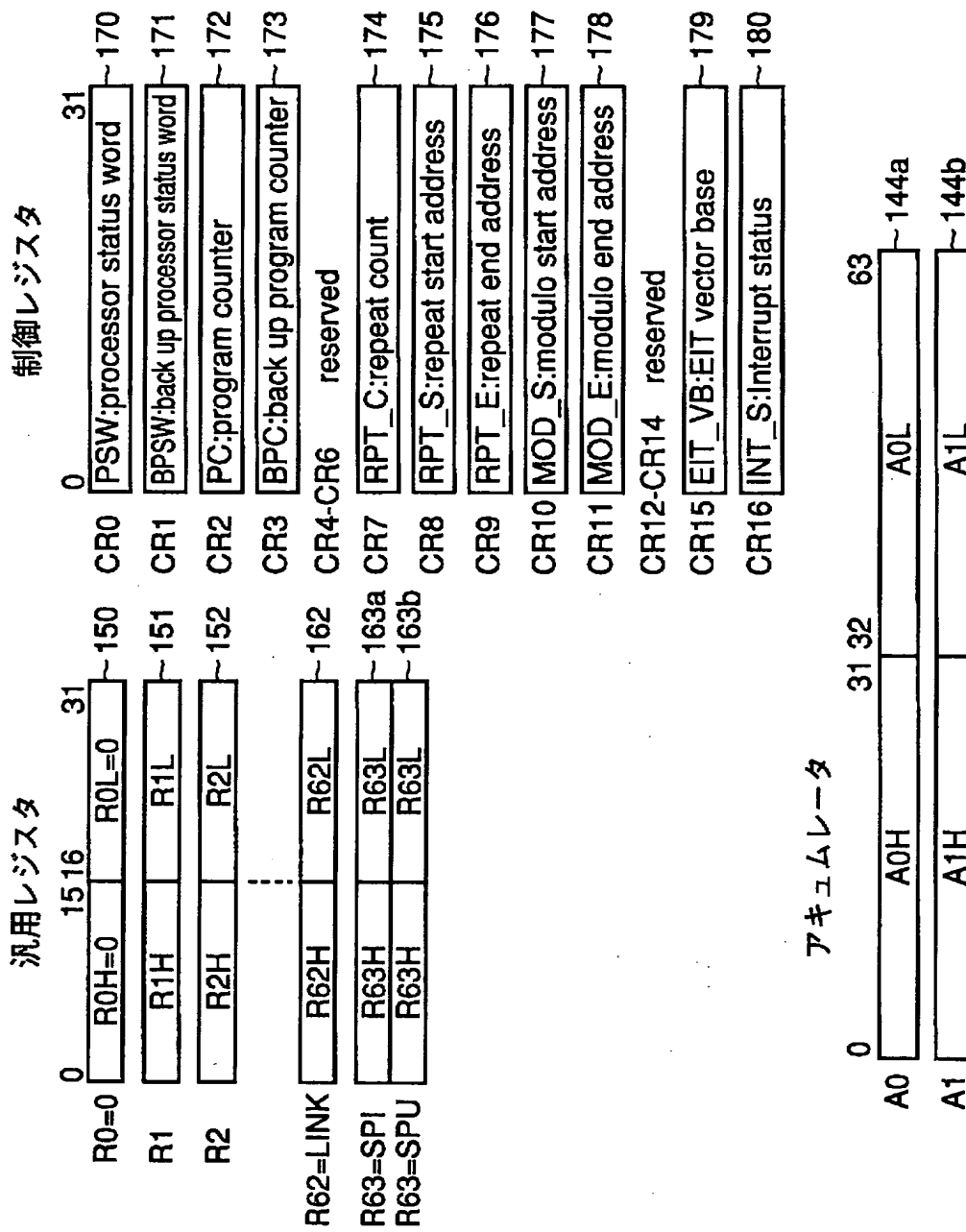
【図 1】



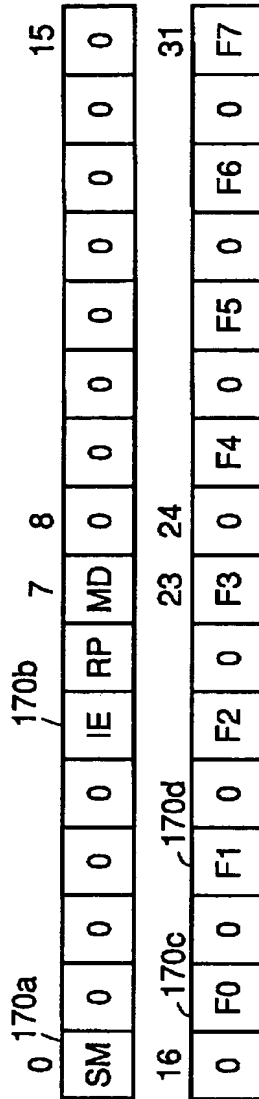
【図 2】



【図 3】

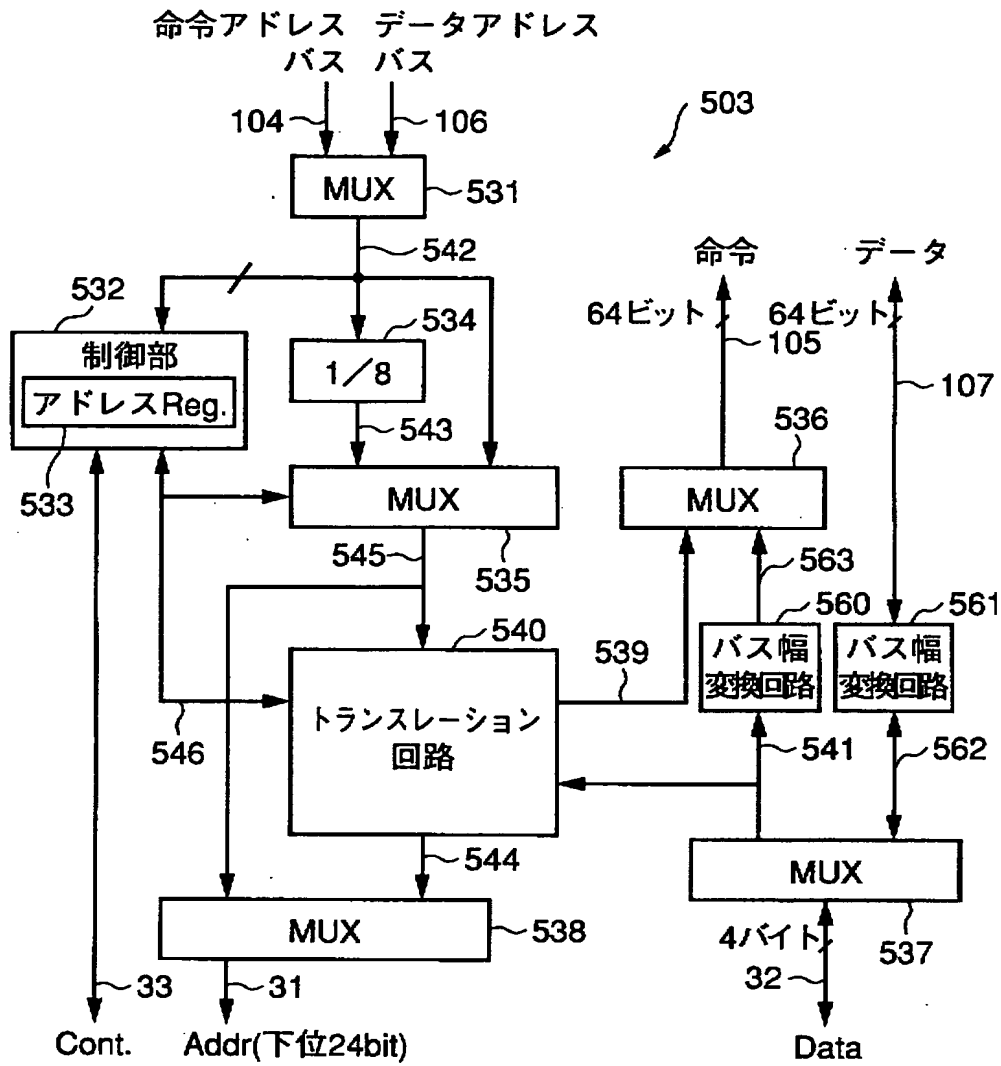


【图 4】

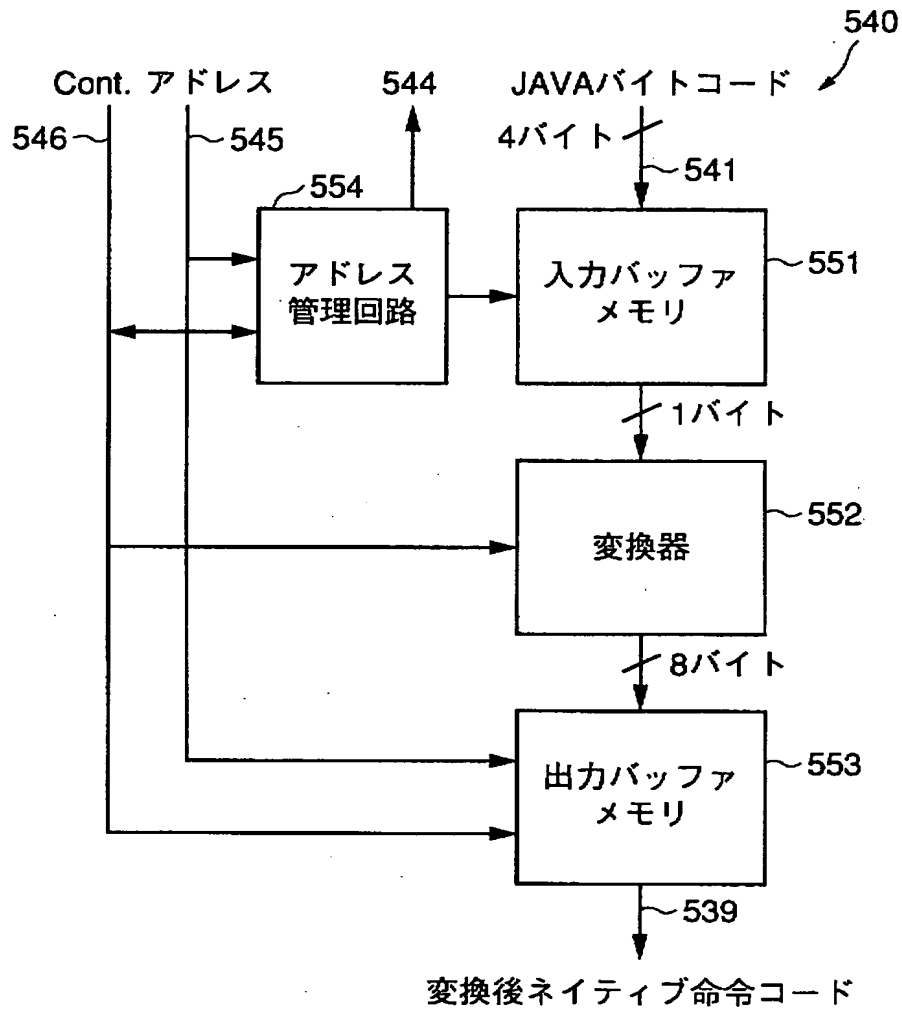


- SM:Stack mode
SM=0 Interrupt mode.SPI is used.
SM=1 User mode.SPU is used.
- IE:Interrupt enable
IE=0 Interrupts are masked.
IE=1 Interrupts are accepted.
- RP:Repeat enable
RP=0 A block repeat is inactive.
RP=1 A block repeat is active.
- MD:Modulo enable
MD=0 Modulo addressing is disabled.
MD=1 Modulo addressing is enabled.
- F0 :Execution control flag #0
- F1 :Execution control flag #1
- F2 :General flag
- F3 :General flag
- F4(S) :Saturation flag
- F5(V) :Overflow flag
- F6(VA) :Accumulated Overflow flag
VA is cleared by a reset interrupt and MVTSYS instruction.
- F7(C) :Carry/Borrow flag

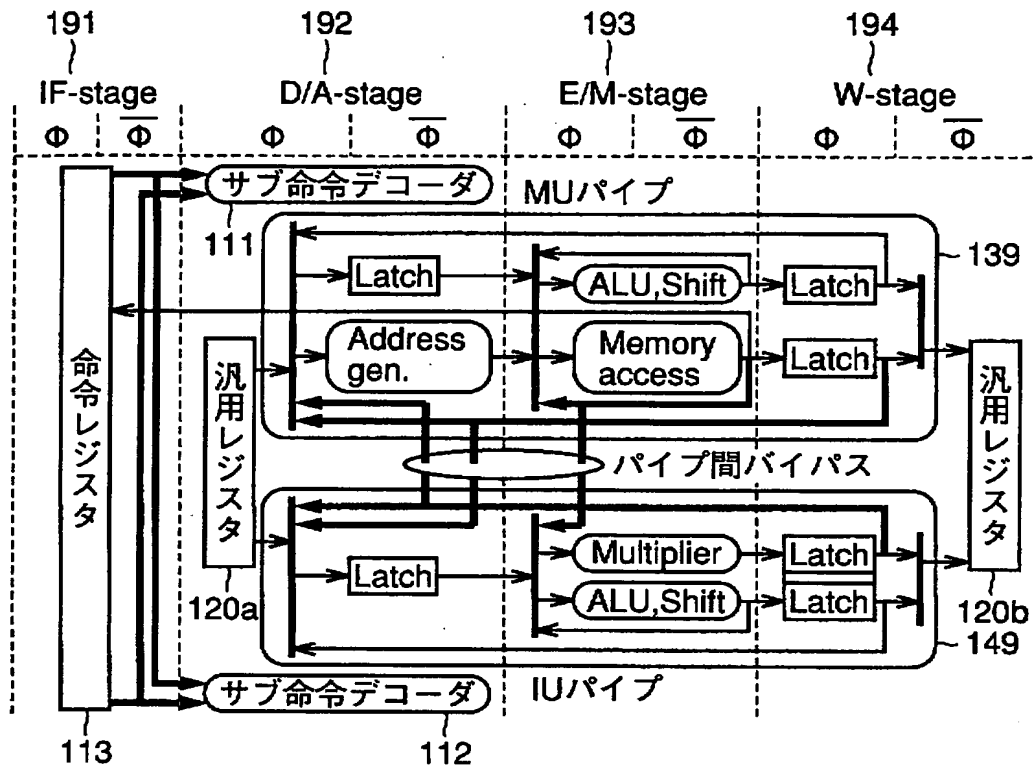
【図 5】



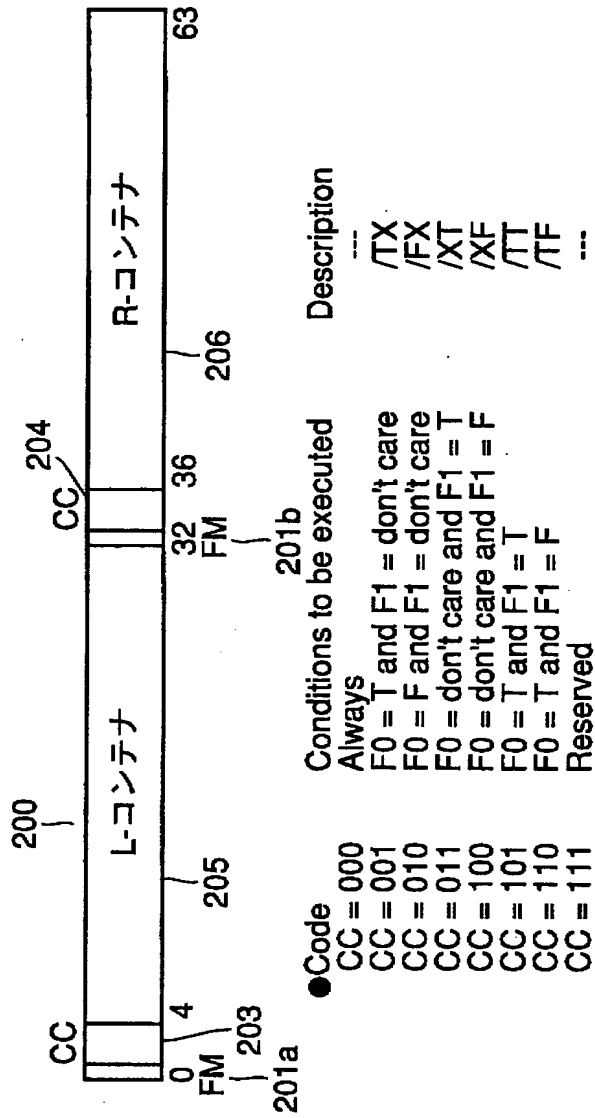
【図 6】



【図 7】

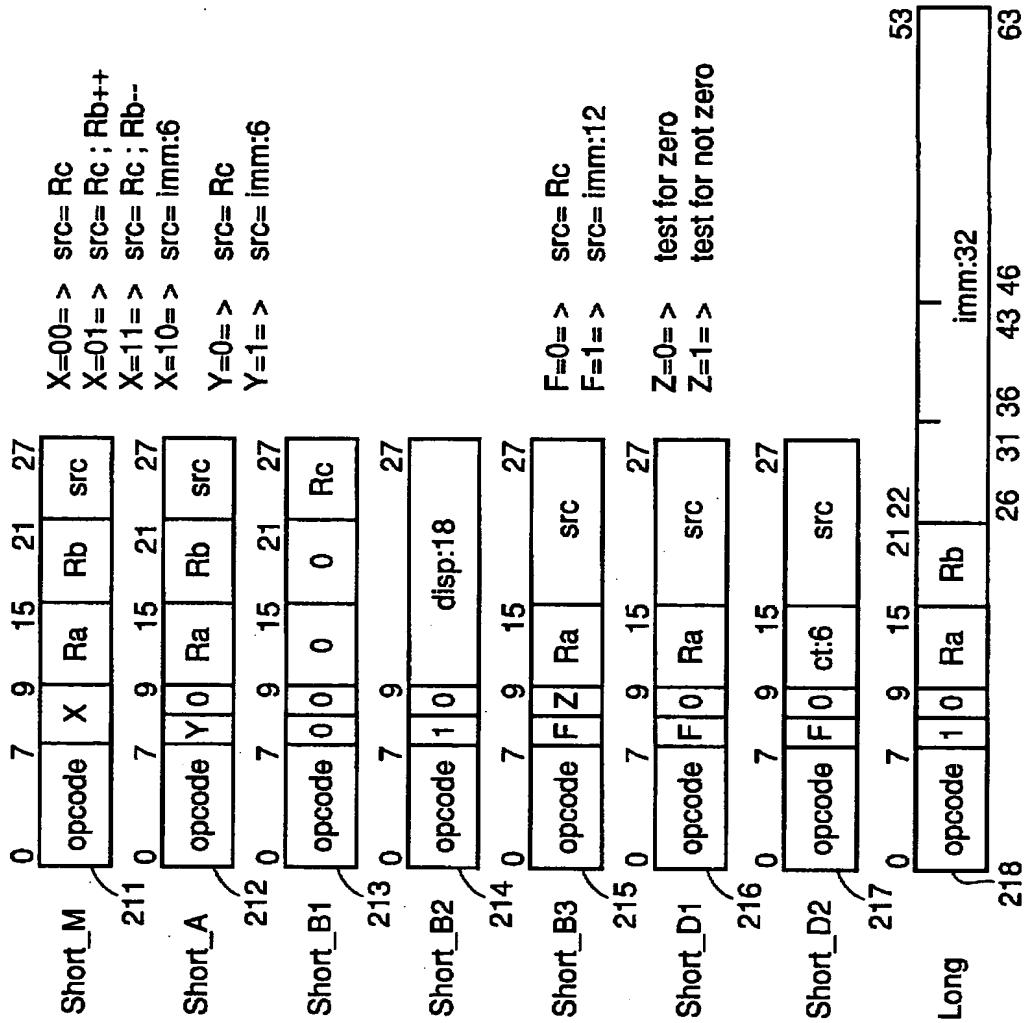


【図 8】

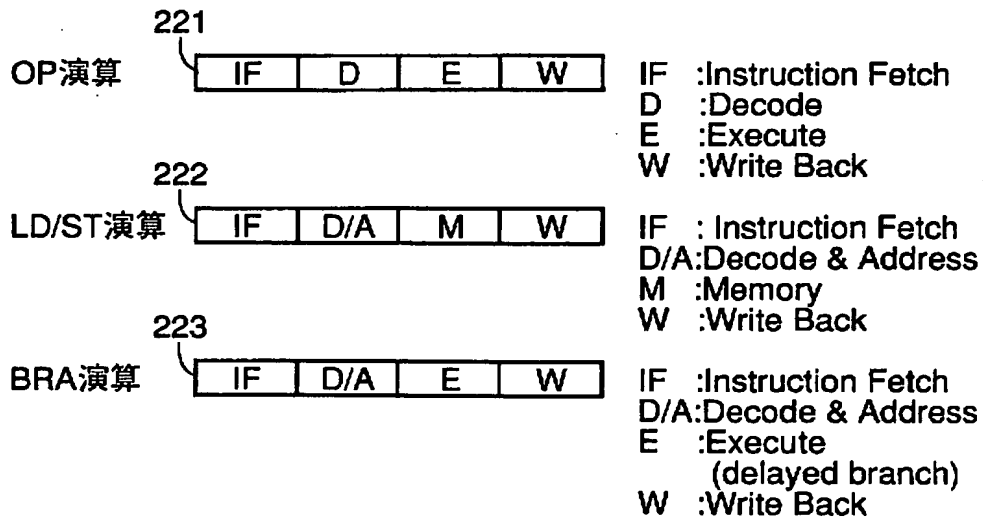


FM	サブ命令の 発行数	発行順序	
		L-コンテナ	R-コンテナ
00	two	1st	1st
01	two	1st	2nd
10	two	2nd	1st
11	one	1st	----

【図 9】



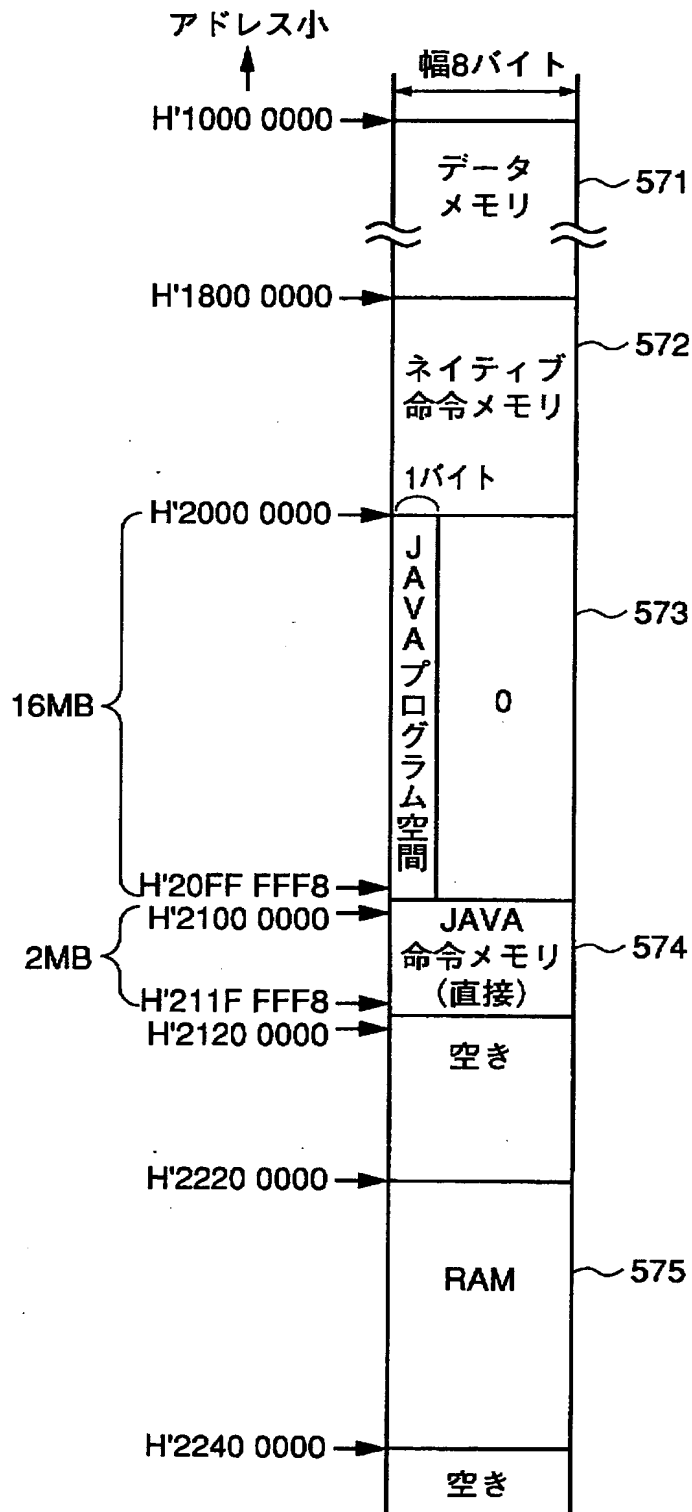
【図 10】



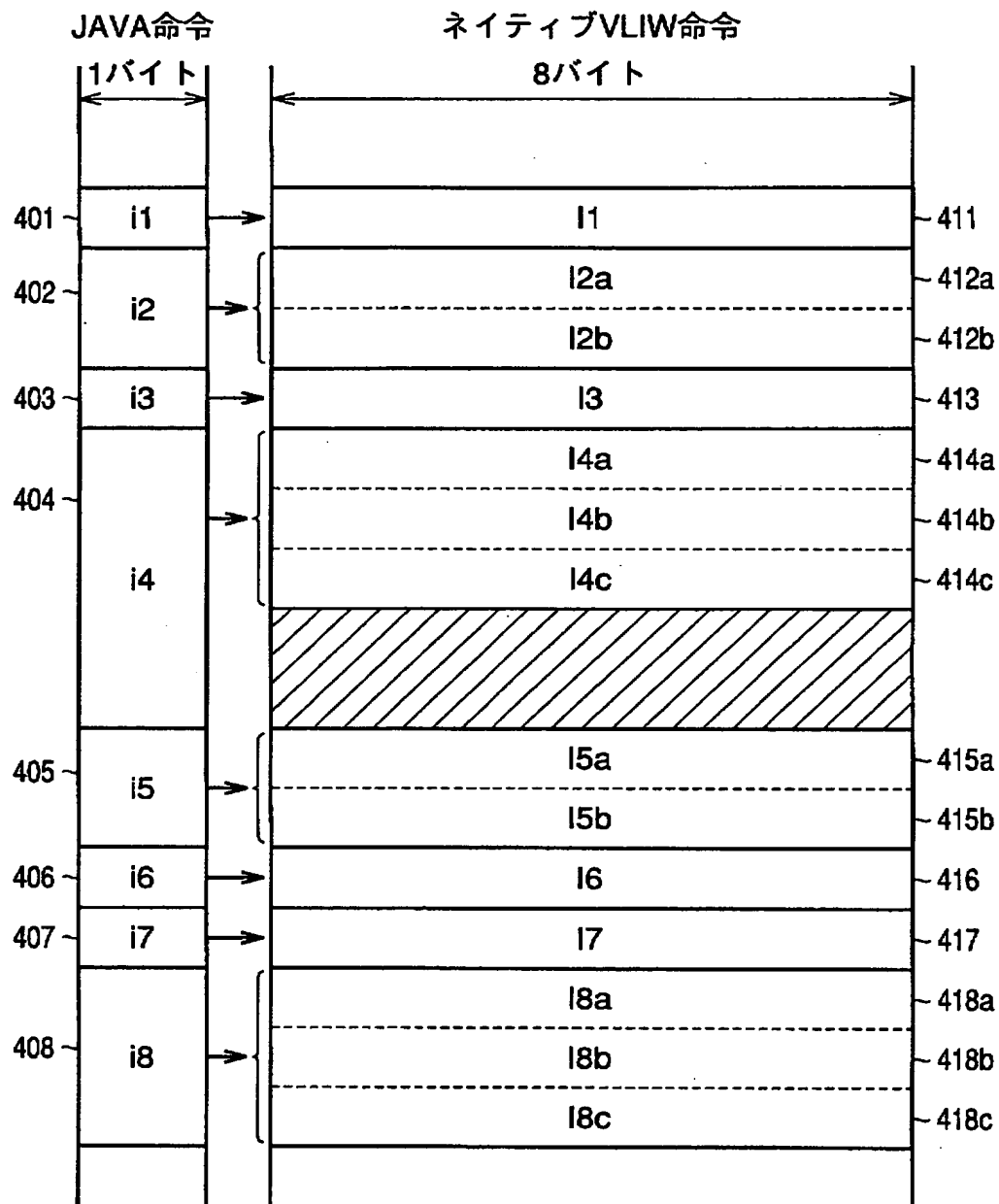
【図 1 1】

FM=00	MU	IF	D or D/A	E or M	W	
	IU	IF	D	E	W	
FM=01	MU	IF	D or D/A	E or M	W	
	IU/MU	IF	stall	D or D/A	E or M	W
234 235						
FM=10	MU/IU	IF	stall	D or D/A	E or M	W
	IU	IF	D	E	W	
236						
FM=11		IF	D or D/A	E or M	W	

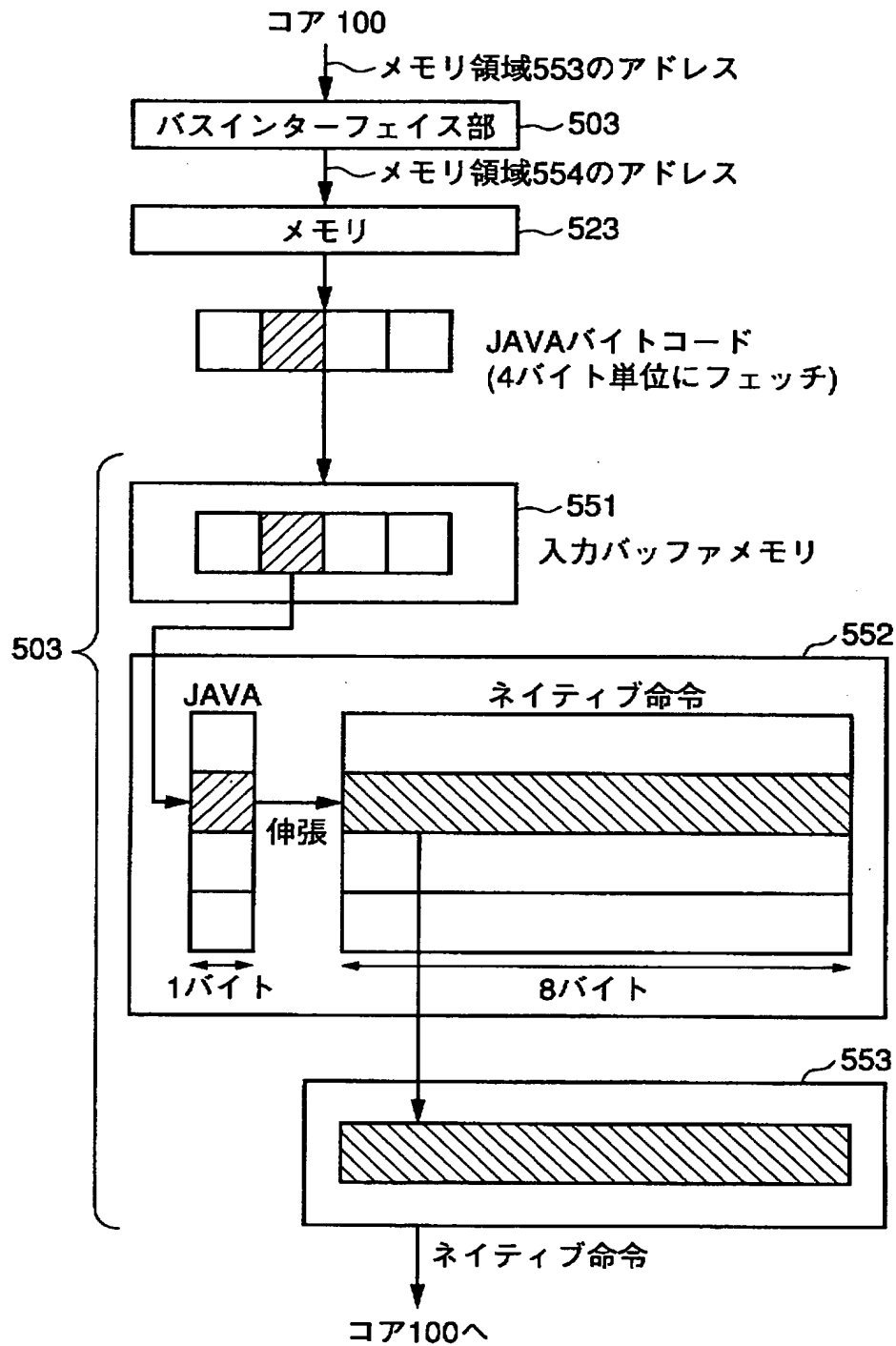
【図 1 2】



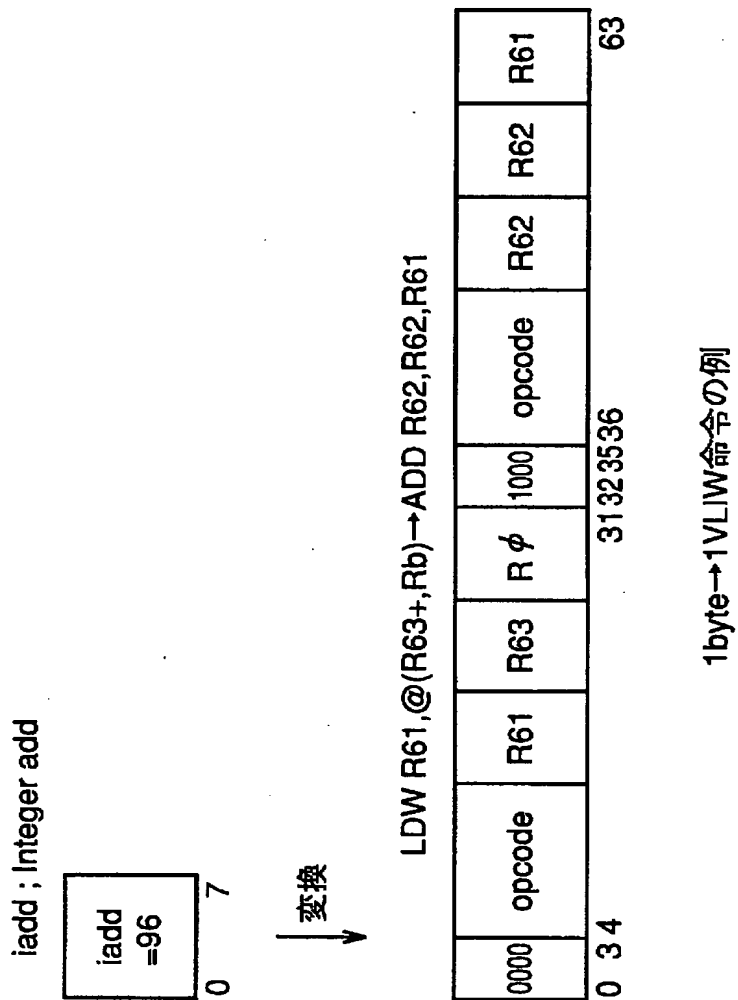
【図 13】



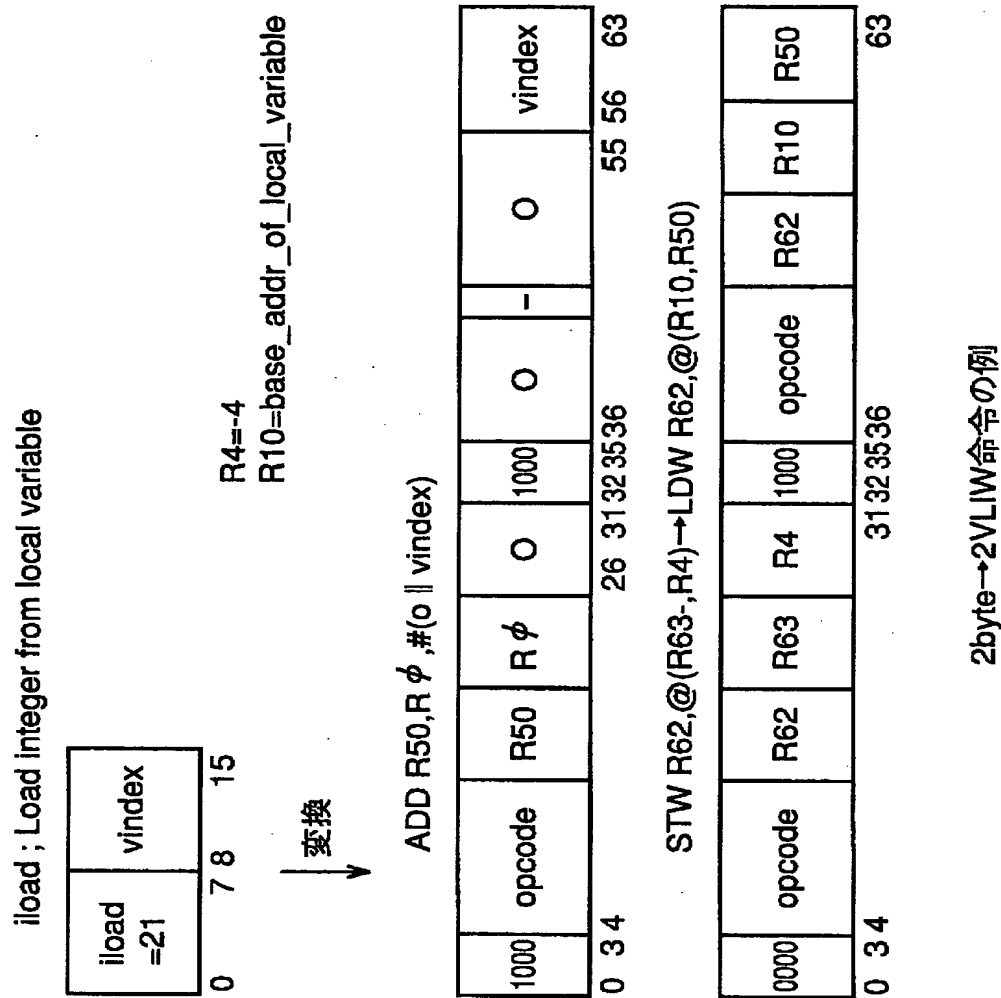
【図14】



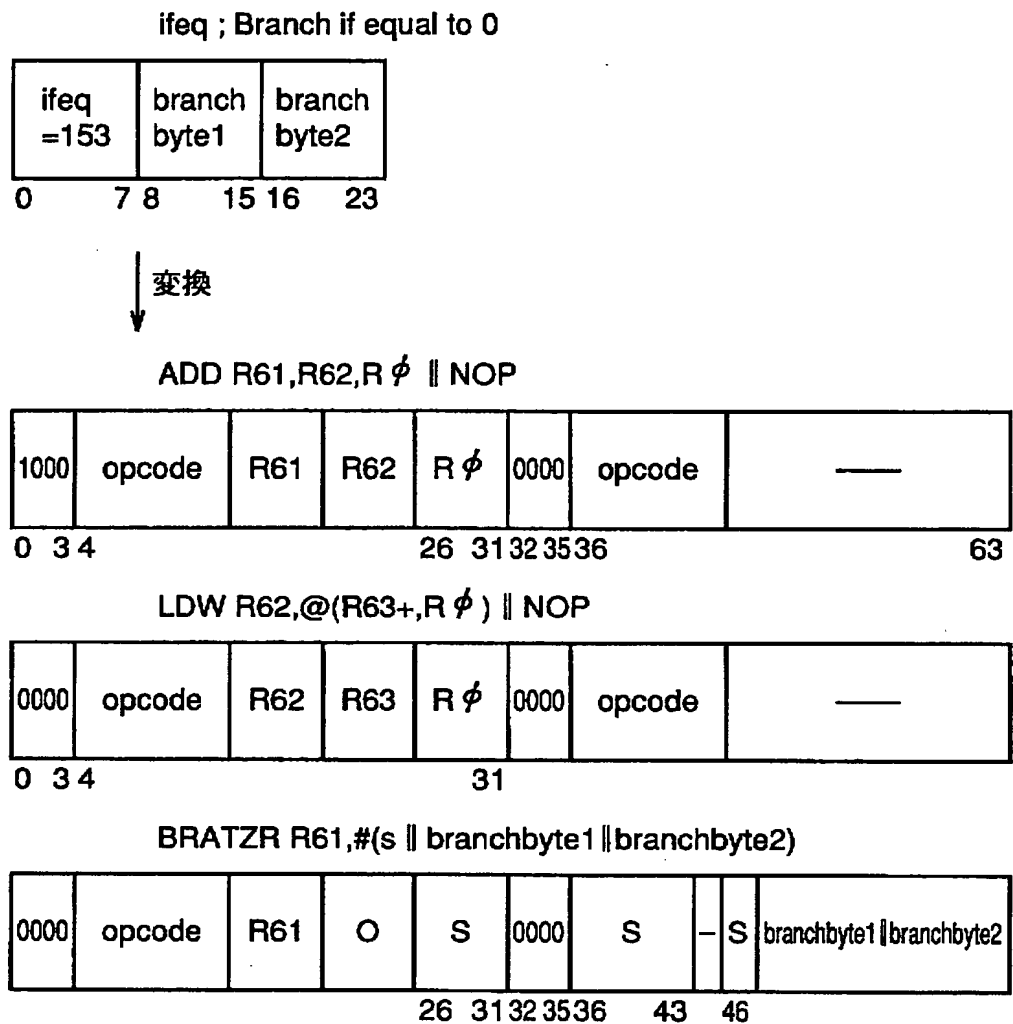
【図 15】



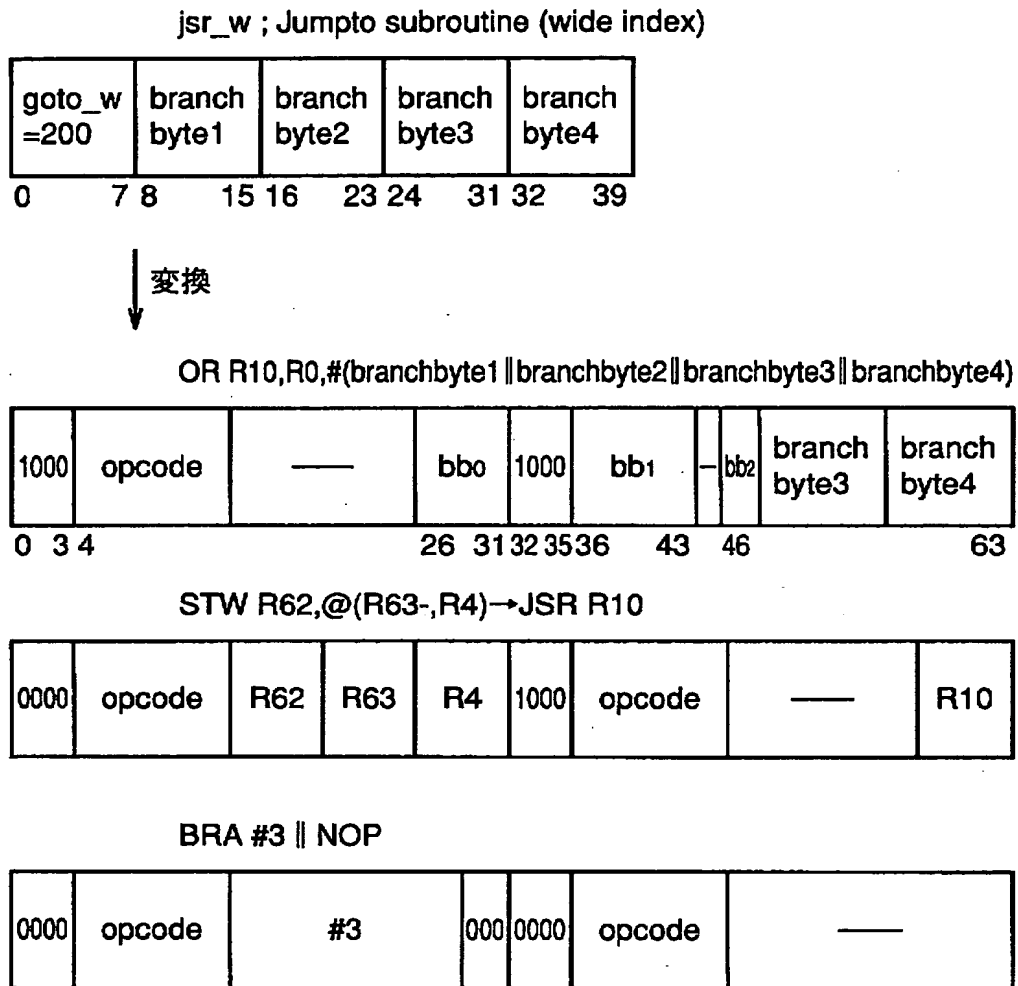
【図 16】



【図 1 7】

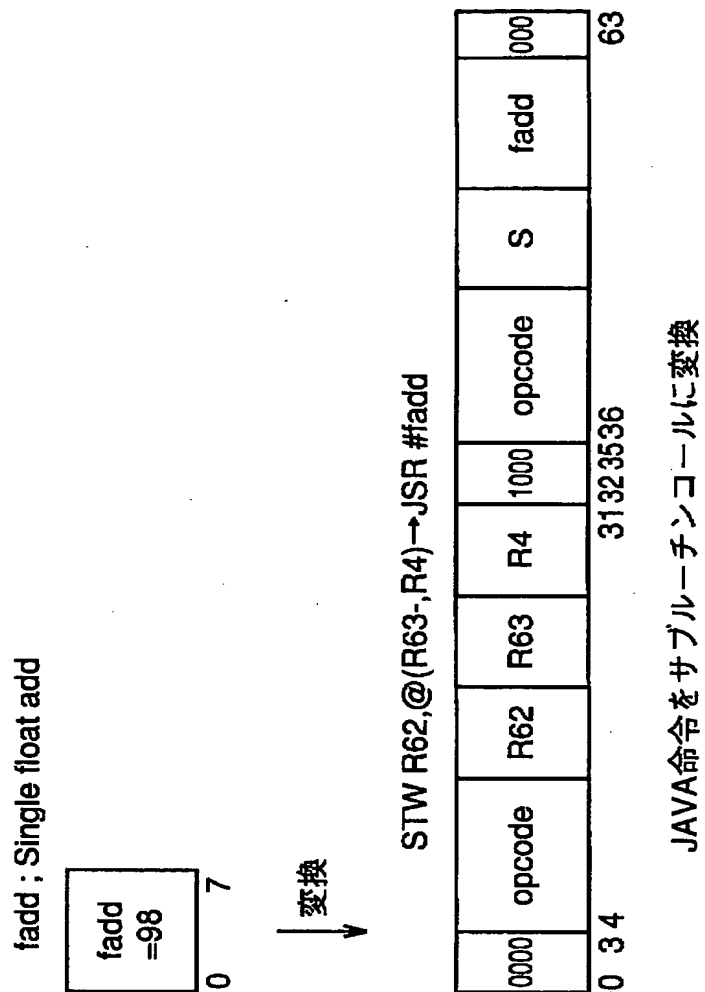


【図 1 8】

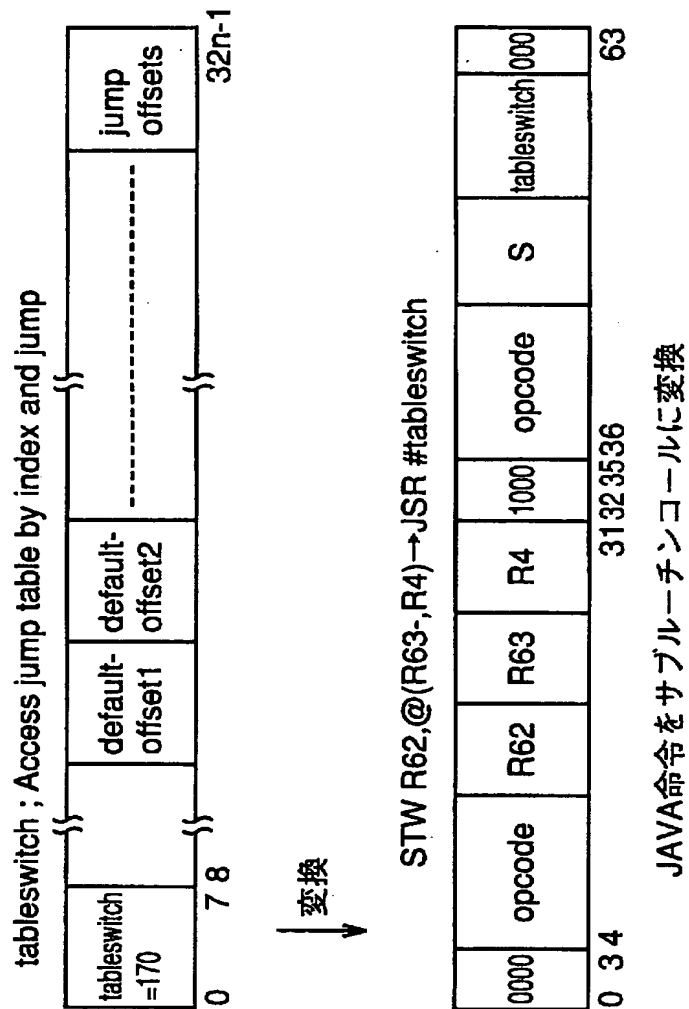


JAVA命令バイト数>VLIW命令数

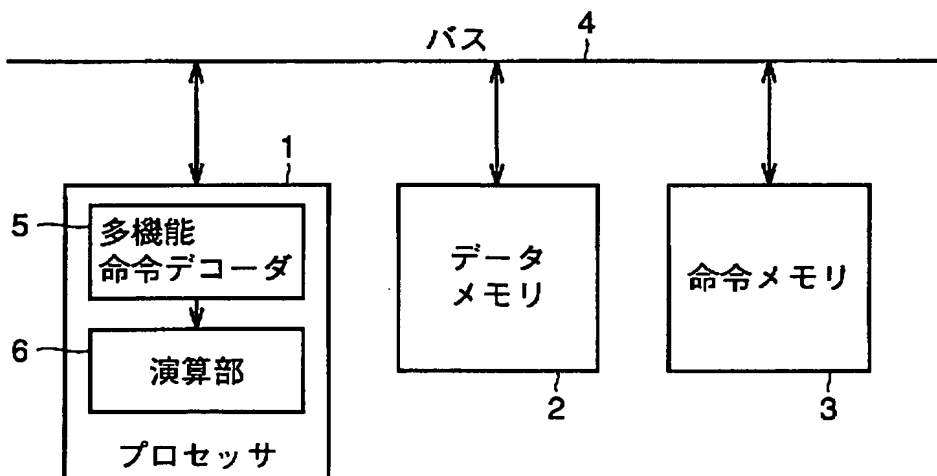
【図 19】



【図 20】



【図 21】



出 願 人 履 歴 情 報

識別番号 [000006013]

1. 変更年月日 1990年 8月24日
[変更理由] 新規登録
住 所 東京都千代田区丸の内2丁目2番3号
氏 名 三菱電機株式会社

【書類名】 要約書

【要約】

【課題】 複数個の異なる命令体系の命令からなるプログラムを高速に実行可能なデータ処理装置とそのためのメモリインターフェース装置を提供する。

【解決手段】 データ処理装置 5 1 0 のメモリインターフェース部 5 0 3 は、プロセッサコア 1 0 0 から外部メモリ空間へのアクセスのためのアドレス値を受け、外部メモリから命令をフェッチするための命令フェッチ回路と、外部メモリから受ける、プロセッサコア 1 0 0 に対する非ネイティブ命令をネイティブ命令に変換するための変換器と、プロセッサコア 1 0 0 から外部メモリ空間へのアクセス時のアドレス値が所定領域内にあるか否かに依存して、外部メモリ空間から読出された命令と、外部メモリ空間から読み出された命令を変換器で変換した結果の命令とのいずれかを選択的にプロセッサコア 1 0 0 に与えるための選択回路とを含む。

【選択図】 図 2